JavaScript 辞典

(株)アンク 著

第4版
fourth edition

HTML5対応

SE
SHOEISHA

最新規格対応の
増補改訂版!

いまやWebにとって欠かせない基幹技術になったJavaScript。従来はWebページに簡単な動きをつけるなど限定的な用途にとどまっていましたが、HTML5の登場によって、ドキュメントの制御、よりインタラクティブなUI、動画や音楽、Webアプリ、スマートフォンアプリなど、高度な実装が行なえるようになりました。

本書『JavaScript辞典 第4版』は、やりたいことから引ける目的別分類とフルカラーの見やすい紙面はそのままに、Canvas、メディア要素、ファイル操作、オフライン処理、位置情報など、モダンなWebサイトやアプリ制作に必須のHTML5 API対応項目を大幅に増補しました。IE10など最新ブラウザやiOS / Android環境の対応状況も掲載しています。

基礎文法の解説や、オブジェクト別のイベント・プロパティ・メソッド一覧も収録しており、紙面のサンプルソースはWebからダウンロードが可能です。学習でも現場でも使える、お役立ちの1冊です。

JavaScript 辞典

HTML5 対応

(株)アンク 著

本書内容に関するお問い合わせについて

このたびは翔泳社の書籍をお買い上げいただき、誠にありがとうございます。弊社では、読者の皆様からのお問い合わせに適切に対応させていただくため、以下のガイドラインへのご協力をお願い致しております。下記項目をお読みいただき、手順に従ってお問い合わせください。

●ご質問される前に

弊社Webサイトの「正誤表」をご参照ください。これまでに判明した正誤や追加情報を掲載しています。

正誤表　　http://www.shoeisha.co.jp/book/errata/

●ご質問方法

弊社Webサイトの「出版物Q＆A」をご利用ください。

出版物Q&A　http://www.shoeisha.co.jp/book/qa/

インターネットをご利用でない場合は、FAXまたは郵便にて、下記"翔泳社 愛読者サービスセンター"までお問い合わせください。

電話でのご質問は、お受けしておりません。

●回答について

回答は、ご質問いただいた手段によってご返事申し上げます。ご質問の内容によっては、回答に数日ないしはそれ以上の期間を要する場合があります。

●ご質問に際してのご注意

本書の対象を越えるもの、記述個所を特定されないもの、また読者固有の環境に起因するご質問等にはお答えできませんので、予めご了承ください。

●郵便物送付先およびFAX番号

送付先住所　〒160-0006　東京都新宿区舟町5

FAX番号　03-5362-3818

宛先　(株)翔泳社 愛読者サービスセンター係

※本書に記載されたURL等は予告なく変更される場合があります。

※本書の対象に関する詳細はxivページの「本書の動作環境」をご参照ください。

※本書の出版にあたっては正確な記述につとめましたが、著者や出版社などのいずれも、本書の内容に対してなんらかの保証をするものではなく、内容やサンプルに基づくいかなる運用結果に関してもいっさいの責任を負いません。

※本書に掲載されているサンプルプログラムやスクリプト、および実行結果を記した画面イメージなどは、特定の設定に基づいた環境にて再現される一例です。

※本書に記載された内容は、2012年12月段階で策定された最新の仕様と、2013年5月現在のブラウザ対応状況にもとづいて執筆されています。仕様、ブラウザ対応状況、その他は今後も変更されることが予想されます。ご了承ください。

CONTENTS

目　次

<ドキュメント>

<ウィンドウ>

<スクリーン>

〈フォーム〉

〈イベント〉

〈非同期通信〉

〈図形とメディア〉

〈ファイル操作〉

INTRODUCTION

本書の読み方

本書は第1部「基礎知識」、第2部「リファレンス」、第3部「オブジェクト一覧」の3部構成となっています。

第1部「JavaScriptの基礎知識」では、初めてJavaScriptを使ってWebサイトを作る人でもこ

【解説ページ】

カテゴリー▶
効果や用途によって分類しています。

タイトル▶
「～したい」という形式で具体的に何ができるのかを表しています。使用目的から選んでください。

値▶
基本書式で★や◆などの記号で表した値の意味です。

形式▶
解説している基本書式の形式（オブジェクト、プロパティ、メソッド、イベント）です。

解説▶
その項目の解説本文です。

文例▶
解説しているプロパティ、メソッド、イベントの使用例です。

コラム▶
その項目に関する注意点や関連するトピック、さらに理解を深めるための内容を紹介しています。

参照▶
その項目と関係の深い項目の参照先です。参照することで体系的に理解できます。

基本書式▶
タイトルで表している内容を表現するための基本的な書式です。解説しているプロパティやメソッド、イベントなどは赤色、引数や親オブジェクトは青色で表記しています。

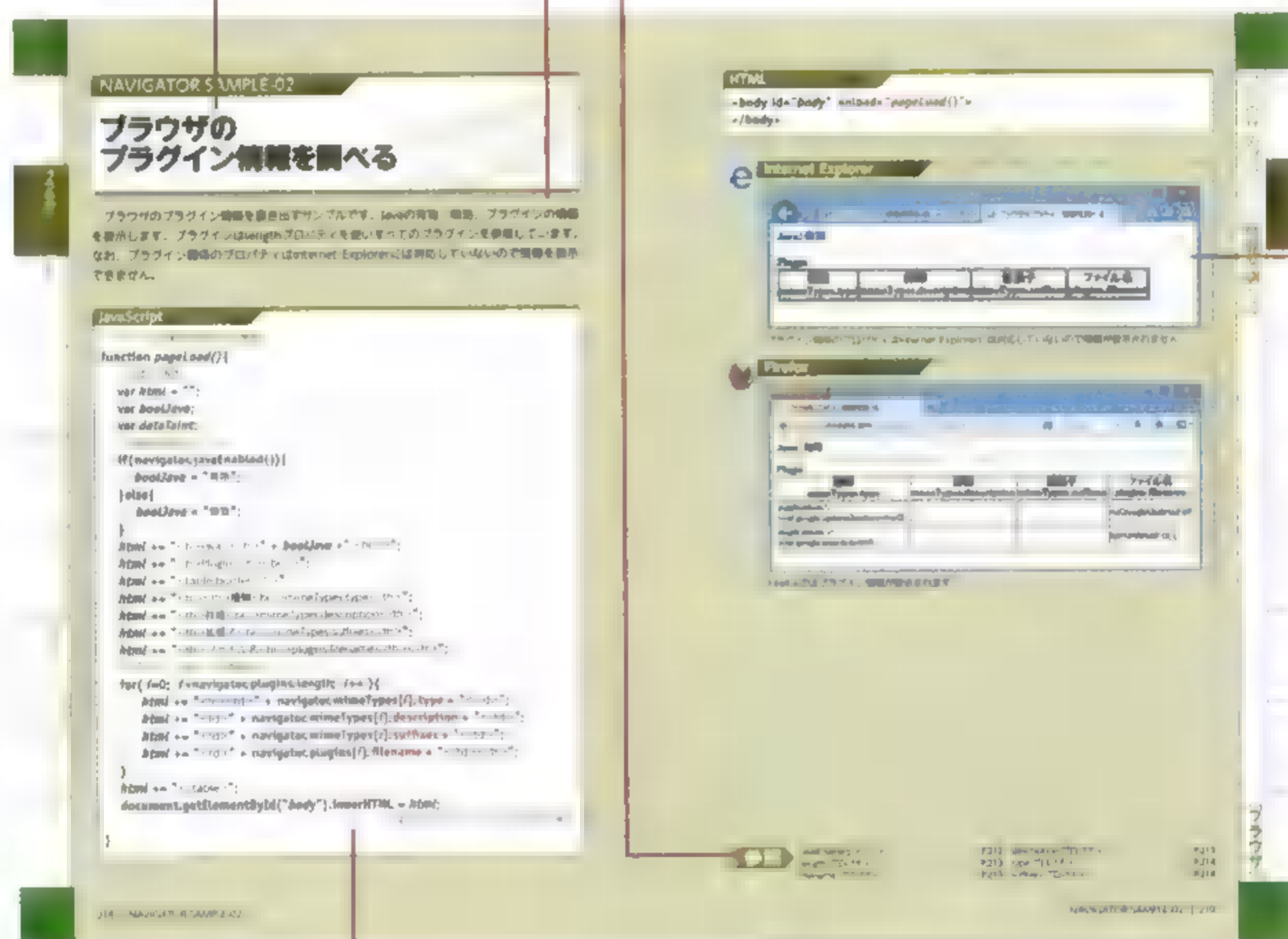
ブラウザ対応表▶
各ブラウザでの対応状況を○×で示しています。掲載しているすべての書式で対応が同じ場合は1行で、異なる場合のみ書式ごとに対応を表記しています。△の場合は注記を掲載しています。

のパートをしっかり読めばJavaScriptを理解できるように、基本的な文法から説明しています。
第2部「JavaScriptリファレンス」では、JavaScriptの効果や利用する場面に合わせて26のカテゴリに分けて各プロパティ、メソッド、イベントなどを解説しています。カテゴリは実際の効果や用途を重視して分類しているため必ずしもオブジェクトごとの分類になっていません。オブジェクトごとのスクリプトを調べるには第3部「オブジェクト一覧」を参照してください。
各カテゴリは解説ページとサンプルソースページで構成されています。解説ページのタイトルは、目的からJavaScriptの機能を引ける形式になっています。内容は基本書式、解説、文例、コラムなどで構成されています。
サンプルソースページでは、解説ページで紹介しているプロパティやメソッド、イベントを使用したサンプルプログラムのソース、解説、ブラウザ表示画面を掲載しています。

【サンプルソースページ】

タイトル▶
そのサンプルでできることを表しています。

解説▶
サンプルソースの解説です。

参照▶
そのサンプルソースで使用しているプロパティ、メソッド、イベントの解説ページの参照先です。

サンプルソース▶
各種プロパティ、メソッド、イベントなどを使用した具体的なサンプルソースです。原則的にJavaScriptソースとHTMLソースの2つからなっています。ソースコードの色分けは解説ページの基本書式に準じ、コメントは緑色で表記しています。また、HTMLソースの中でJavaScriptの呼び出しに使われている部分は赤字です。ユーザーが任意で名前を付けられる関数名や変数名は斜体で表記しています。

サンプル表示画面▶
ソースを掲載したサンプルプログラムを実際にブラウザで表示したときの画面です。基本的にはInternet Explorer 10での表示画面を掲載し、一部の項目でFirefoxなどほかのブラウザの画面を掲載しています。iPhoneやAndroidスマートフォンのブラウザ画面を掲載している場合もあります。

INTRODUCTION

本書の動作環境

■OSとブラウザの[illegible]

本書は、以下の[illegible]におけるブラウザ表示にもとづいて記述しており、リファレンス内の対応表も以下の環境での結果となります。

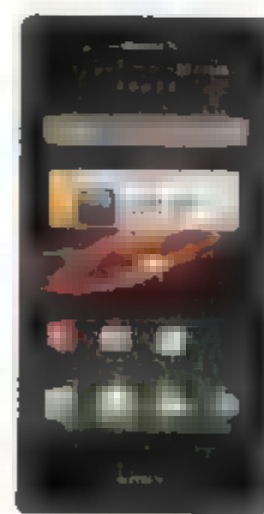
OS	日本語版 Microsoft Windows 8/7/Vista/XP
ブラウザ	Internet Explorer10/9/8、Firefox 21.0、Google Chrome 26、Opera12.15
OS	日本語版Mac OS X 10.8.2
ブラウザ	Firefox 21.0、Google Chrome 26、Opera12.15、Safari 6.0.2
OS	iOS 6.1.3
ブラウザ	iPhone Safari
OS	Android 4.1
ブラウザ	Android標準ブラウザ

※Windows版Safariは2012年7月に提供が終了したため、SafariはMacでの[illegible]結果となります

■検証用[illegible]

本書の検証に使用したモバイル端末は以下の通りです。

- Apple iPhone 5/4(iOS 6.1.3搭載)
- Xperia Z SO-02E(Android 4.1搭[illegible])

Apple iPhone 5

Xperia Z SO-02E
協力:NTTドコモ

■ディスプレイ[illegible]

サンプルソースを表示しているディスプレイ画面は、[illegible]本的に各ブラウザの最新バージョン(Internet Explorerなら10)の初期設定のものを掲載しています。効果が明確に現れるように、適宜画面表示を拡大している場合もあります。

掲載画面はあくまでも一例ですので、ブラウザの設定やお使いのセキュリティソフトの設定によっては、本書の表示通りにはならないので注意してください。

■サンプルデータ

本書のサンプルデータは以下のURLよりダウンロードまたは、[illegible]のページのQRコードよりアクセスしてください。

http://www.shoeisha.com/book/pc/dic/

第1部
JavaScriptの基礎知識

JavaScript BASIC

BASIC.01

JavaScriptとは

JavaScriptの特徴

JavaScriptはWebブラウザ上などで動作するスクリプト言語(簡易プログラミング言語)です。

JavaScriptの身近な利用例としては、クリックしたときにアラートを表示する、任意の設定で新しいウィンドウを開き、新しいウィンドウから元のウィンドウの内容を操作するといったものがあります。HTML/XHTMLだけで作成されたWebページは静的な表現しかできませんが、JavaScriptを利用することでページに動きを持たせることが可能になったのです。

開発当初はWebブラウザで利用することを目的としていましたが、現在ではWebサーバ上でWebページを作成するため、あるいは汎用的なスクリプト言語としてWebページ以外の分野でも広く用いられるようになっています。

JavaScriptは後述するように、主にHTML/XHTML文書中のscript要素の内部に記述するか、スクリプトを記述したファイルを別に保存しておき(拡張子「.js」)、保存したファイルを読み込むことで動作させることができます。コンパイル(プログラミング言語で作成したソースコードをコンピュータが理解できる[illegible]語に変換すること)作業を必要としないインタープリタ言語のため、JavaScriptに対応したブラウザがあれば手軽にスクリプトを実行できるという特徴を持っています。

JavaScriptの歴史

JavaScriptはNetscape社が開発したLiveScriptが原型になっており、その後Sun Microsystems社との共同開発の流れを受けてJavaScriptと名称を変更し、現在に至ります。

JavaScriptはまずNetscape Navigator 2.0に搭載され、その後、Microsoft社のInternet Explorer 3.0にも搭載されました(正確にはInternet Explorerに搭載されているのはJavaScript互換のJScript)。しかし、ブラウザ間の実装に若干の違いがあり、ブラウザによって使えない[illegible]があったり、同じプログラムでも動作が異なったり、といった問題が生じたため、ヨーロッパの標準化団体ECMAが両社へ呼びかけ、JavaScriptの中核的な仕様がECMAScriptとして標準化されました。この標準化によってJavaScriptは多くのブラウザで利用できるようになったのです。Microsoft社のJScriptの例に見られるように独自拡張などを施されるケースも多く、その場合は独自の名称を付けることが慣習になっているようです。

BASIC.02

JavaScriptの組み込み方

JavaScriptをWebページに組み込むには次の方法があります。

・HTML/XHTML文書内に記述する方法
・外部ファイルに記述して読み込む方法
・その他の方法(HTML/XHTML要素内に直接記述する方法、ブラウザで直接実行する方法)

HTML/XHTML文書内に記述する方法

<script>タグで囲んだ中にスクリプトの内容を記述します。type属性には記述する言語のMIMEタイプを指定します。HTML5ではtype属性は省略可能で、text/javascript(JavaScriptのMIMEタイプ)が既定値となりました。

```
<script type="text/javascript">★</script>
★……スクリプトの内容
```

※HTML5ではtype属性省略可能

Column language属性

従来はtype属性ではなく、language属性を利用してスクリプト言語の指定を行っていましたが、language属性はHTML 4.01/XHTML 1.0では「非推奨」に指定されているため使用しません(Transitional DTDかFramesetDTDであれば利用できます)。HTML5では仕様から除外され、「強く非推奨」となりました。

Source ※紙面の都合上、文書型宣言などは省略しています。

```
<!DOCTYPE html>
<html>
<head>
<title>JavaScript Sample</title>
<script>
<!--
ここにスクリプトを記述します。
// -->
</script>
</head>
<body>
<script type="text/javascript"> // JavaScriptは複数記述できる
<!--
```

```
ここにスクリプトを記述します。
    /*特定の場所に書き出すには<body>タグの中の
    書き出したい場所に記述する*/
// -->
</script>
<noscript>
    <p>このページはJavaScript対応ブラウザで見てください。</p>
</noscript>
</body>
</html>
```

■スクリプト内容のコメント化

script要素内にスクリプトを書く場合、スクリプト全体を「<!--」と「//-->」で囲み、古いブラウザなどでスクリプトの内容がそのまま表示されるのを防ぐのが一般的でした。

```
<script type="text/javascript">
<!--
ここにスクリプトを記述します。
//-->
</script>
```

XHTML 1.0の仕様ではこのように記述するとコメントとして解釈され、無視されることになっているため、代わりに「//<![CDATA[」と「//]]>」で囲って記述します。HTML5では、XHTML書式の場合はこれと同様に記述しますが、HTML書式の場合は従来のコメント化でも構いません。

```
<style type="text/javascript">
//<![CDATA[
ここにスクリプトを記述します。
//]]>
</style>
```

しかし、最新でないバージョンのブラウザではこの方法がサポートされていないことがあります。こうした問題に対処するためには次のような方法があります。

●外部ファイルにする(p.005)

スクリプトだけを記述したファイルを別に用意し、HTML/XHTML文書から読み込む方法です。外部ファイルではスクリプトをコメント化する必要はありません。XHTMLでは、この方法が奨励されています。

●スクリプト内容をコメント化しない

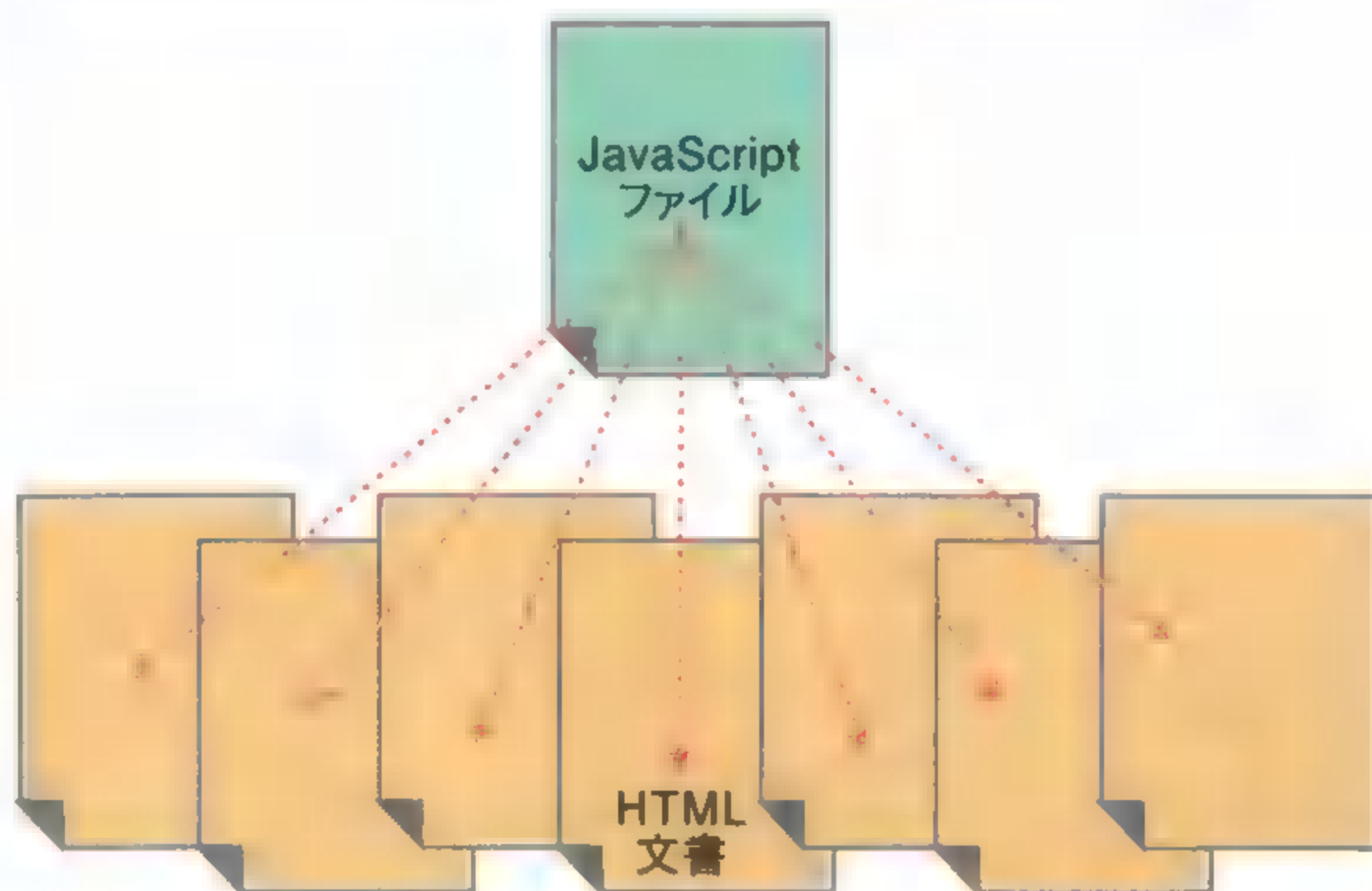
script要素内に直接スクリプトを記述します。script要素に未対応の古いブラウザなどでは、スタイルの記述箇所がそのまま表示されてしまう可能性もありますが、現在一般的なブラウザ

はどれもscript要素に対応しているので大きな問題はないでしょう。

また、現在一般的なブラウザでは、XHTML文書でHTML文書と同様に「<!--」と「-->」で囲ってコメント化しても問題なく実行されるようですが、これはXHTML的には文法的に正しい書き方とは言えません。なるべく仕様の奨励に従って、コメント化の不要な外部ファイルを利用するのがよいでしょう。

外部ファイルに記述して読み込む方法

スクリプトだけを記述したファイルを別に用意し（拡張子は「.js」）、script要素のsrc属性でファイル名（URI）を指定して読み込む方法です。script要素の内容には何も書かずに、終了タグ</script>を付けます。XHTMLではこの方法が奨励されています。

```
<script type="text/javascript" src="★"></script>
★……スクリプトファイル名（URI）
```

外部ファイルを使えば、1つのJavaScriptファイルを複数のHTML文書で使用できます

Source

```
document.write("Hello");
```

Source

```
<html>
<head>
<title>JavaScript Sample</title>
<script type="text/javascript" src="hello.js"></script>
</head>
<body>
```

```
<script type = "text/javascript" src="konnichiwa.js"></script>
<noscript>
    <p>このページはJavaScript対応ブラウザで見てください。</p>
</noscript>
</body>
</html>
```

その他の方法

■HTML/XHTMLの要素内に直接記述する

HTML/XHTMLの要素の属性(イベント属性)に設定する方法です。

■ブラウザで直接実行する

HTML/XHTMLのa要素のhref属性、またはWebブラウザのアドレスバーにjavascript:~という形でスクリプトを記述し、実行する方法です。

たとえば、次の一文をアドレスバーに入力して実行した場合、「アドレスバーから実行しています」というダイアログが表示されます。

```
例：javascript:alert("アドレスバーから実行しています");
```

また、次の例ではクリックしても何も動作しないようになります。voidは何も値を返さない命令です(p.246)。

```
例：<a href="javascript:void(0)">クリックしても何も起こりません。</a>
```

どの方法で記述するか

現在のWebではHTML/XHTML本来の「文書構造を示す」機能と、それ以外のプレゼンテーションに関わる機能を分離させるようになってきています。外部スクリプトを読み込む方法であればHTML/XHTML文書からスクリプト部分が切り離されるため、こうした方針に従うことができます。また、前述のようなスクリプトのコメント化に関する問題にも対処できるというメリットもあります。そのため、なるべく外部スクリプトを使用して記述するのが望ましいでしょう。本書でも基本的に外部スクリプトを使用しています。

記述する位置

HTML/XHTML文書内に記述する方法、外部ファイルに記述して読み込む方法のどちらの場合も、script要素はHTML/XHTML文書のhead要素内やbody要素内にも記述できます。どこに記述するかは自由ですが、JavaScriptの性質上読み込まれた順番に実行されるので、特定の場所で実行させたい場合にはその位置に記述してください。スクリプトはHTML/XHTML文書の中に複数記述することもできます。

デフォルトのスクリプト言語の指定

HTML/XHTMLはさまざまなスクリプト言語を利用できるため、デフォルトのスクリプト言語を明示しなければなりません。次の一文をhead要素に記述してください。ただし、HTML5では「text/javascript」が既定のため指定不要です。

```
<meta http-equiv="content-script-type" content="★" />
★……スクリプト言語を示すMIMEタイプ(「text/javascript」「text/vbscript」など)
```

※HTML5では省略可

content属性の値にはスクリプト言語を示すMIMEタイプを指定します。JavaScript言語をデフォルトのスクリプト言語とする場合は「content="text/javascript"」としてください。

イベント属性で要素に直接スクリプトを記述する場合にはここで指定した言語として解釈されることになっています。

例：
```
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=Shift_JIS" />
  <meta http-equiv="Content-Script-Type" content="text/javascript" />
  <title>デフォルトのスクリプト言語の設定</title>
      :
</head>
```

コメントの書き方

JavaScriptでは、その行の「//」より後ろに記述された部分はコメントとなります。また、「/*」と「*/」で囲まれた部分もコメントです。「/*」～「 */」を使用する場合はコメントを複数行にわたって記述できます。

コメントの部分は実行時に無視され、スクリプトの動作に影響を与えません。スクリプトに関する説明を記述したり、エラーの原因と思われる部分をコメントアウトして動作を確認したりするなどの用途で利用されます。

本書ではコメントを緑色の文字で掲載しています。

非対応ブラウザへの配慮

noscript要素を使うことで、JavaScriptに対応していない環境でページを開いた人にだけ表示させるHTML/XHTML文書を記述することもできます。HTML5ではhead要素内でも使用できるようになりました。

BASIC.03

JavaScript記述の注意点

JavaScriptの基本書式や記述するときの注意点をまとめておきましょう。

半角文字で記述

JavaScriptでは基本的に半角の英数字と「{}」(中カッコ)、「()」(小カッコ)、「""」(ダブルクォーテーション)などの記号が使用できます。ただし、「""」や「''」(シングルクォーテーション)でくくられた文字はStringオブジェクトとして扱われるため、「""」や「''」でくくれば全角文字も使用できます。

JavaScriptでは大文字小文字が区別されます。たとえば次の例1の処理を例2のように記述すると、エラーになってスクリプトは動作しません。

例1 : **document.write("こんにちは");**
例2 : **document.Write("こんにちは");**

また、変数名や関数名などについても同様で、たとえば変数myDayと変数mydayは別のものとして解釈されます。スペルミスに十分注意してください。

変数名や関数名などのユーザーが任意の名前を指定できるものについては次のような命名規則があり、これに従えば自由に名前を付けることができます。

- 半角のアルファベット(a～z、A～Z)、数字(0～9)、「_」(アンダーバー)、「$」(ドル記号)を使用する
- 1文字目には数字は使用できない
- 予約語でない(右記参照)

予約語とは言語の仕様で既存のキーワードとして予約されているもの、あるいは将来のキーワードとして予約されているものなどで、変数名として使用できません。

予約語一覧						
abstract	boolean	break	byte	case	catch	char
class	const	continue	debugger	default	delete	do
double	else	enum	export	extends	FALSE	final
finally	float	for	function	goto	if	implements
import	in	instanceof	int	interface	long	native
new	null	package	private	protected	public	return
short	static	super	switch	synchronized	this	throw
throws	transient	try	true	typeof	var	void
volatile	while	with				

スクリプト内の改行とセミコロン

JavaScriptは1つまたはいくつかの命令文で処理が構成され、処理の区切りを「;」(セミコロン)で表します。

1行に1つの命令を書くのであれば、セミコロンはなくても動作しますが、区切りを明確にする意味でも付けておいた方がよいでしょう。セミコロンが入っていれば、複数の処理をまとめて1行に記述することができます。以下は、2つの命令を1行にまとめた例です。

例：*today* = new *Date()*; t = today.getDate();

逆に1行の記述が長くなる場合は、途中で改行を入れても動作に影響ありません。ただし、プログラムを構成する単語や文字列の途中で改行を入れることはできません。以下は1つの命令を2行にした例です。

例：
```
document.write("訪問済みリンクの色は"
  + document.vlinkColor + "です。");
```

数値

JavaScriptで扱える数値は整数(1、-2など)と浮動小数点数(2.145など)です。整数では、10進数のほかに16進数、8進数も扱えます。16進数は先頭に0xもしくは0Xを付け、8進数は先頭に0を付けた表記方法で表します。たとえば16進数のffをJavaScriptで扱うには0xffと表記します。

文字列

JavaScriptで文字列を扱う場合は、「"」(ダブルクォーテーション)または「'」(シングルクォーテーション)で囲みます。「"」や「'」で囲まれた場合は数字であっても数値ではなく文字列として扱われます。

また、スクリプト内にタグを埋め込む場合、タグも文字列の一部として「"」や「'」で囲む必要

があります。複数の文字列をつなげるには、文字列連結演算子「+」(プラス)を使用します。

```
例：today = new Date();
    t = today.getDate();
    document.write("こんにちは。いいお天気ですね。<br />");
    document.write('今日は<b>' + t + '</b>日です。');
```

なお、write、writelnメソッドで「,」(カンマ)で区切って文字列を書くと、つながって出力されます。

論理値(ブール値)

論理値は、true(真)またはfalse(偽)のいずれかの値を取り、条件分岐(p.024)などで利用されます。

その他の値

値が何もない状態を示すnull、値が見つからない場合や定義されていないことを示すundefinedなどがあります。

BASIC.04

JavaScriptにおける色の指定

JavaScriptで色を指定するには、HTML/XHTML同様、RGB値を用いる方法と、色名を用いる方法とがあり、それぞれ次のように指定します。

■#rrggbb（16進数で指定）

#に続けて赤(r)、緑(g)、青(b)の値を00～ff の16進数計6桁で表現します。たとえば、黒を指定する場合には#000000となります。基本的な16色については下[illegible]を参照してください。

■色名（色の名前で指定）

色名で直接指定します。大文字と小文字は区別されません。HTML 4.01では基本的な16色が定義されています。基本的な16色については下[illegible]を参照してください。

red	[illegible]	navy	#000080	[illegible]	[illegible]	black	#000000
[illegible]	[illegible]	[illegible]	[illegible]	lime	#00ff00	[illegible]	[illegible]
purple	#800080	aqua	#00ffff	[illegible]	[illegible]	silver	#c0c0c0
maroon	#800000	[illegible]	[illegible]	yellow	#ffff00	white	#ffffff

Column　CSSでの色指定

JavaScriptではCSSのスタイルも操作することが可能です。その場合はCSSの色の指定方法を利用できます。

以下はすべてmidashiというオブジェクトの文字色を赤に設定しています。

```
midashi.style.color = "#ff0000";
midashi.style.color = "#f00";
midashi.style.color = "rgb(100%,0%,0%)";
midashi.style.color = "rgb(255,0,0)";
midashi.style.color = "red";
```

なお、CSS3では、HTML 4.01で定義された基本的な16色に加えて、SVG仕様に準じる多数のキーワードが追加されています。

※CSSの詳しい解説は本書姉妹書「HTML5&CSS3辞典 第2版」を参照ください。

BASIC.05

オブジェクト、プロパティ、メソッド

JavaScriptはオブジェクトベースのスクリプト言語です。ここではJavaScriptを理解するのに不可欠な、オブジェクト、プロパティ、メソッドといった言葉とその[illegible]を説明します。

オブジェクト

JavaScriptでは、ウィンドウ、ドキュメント、フォーム、文字列、日付など、ページの表示に関わるさまざまなものを制御します。操作の対象となるこれらのものはすべてオブジェクトと呼ばれます。

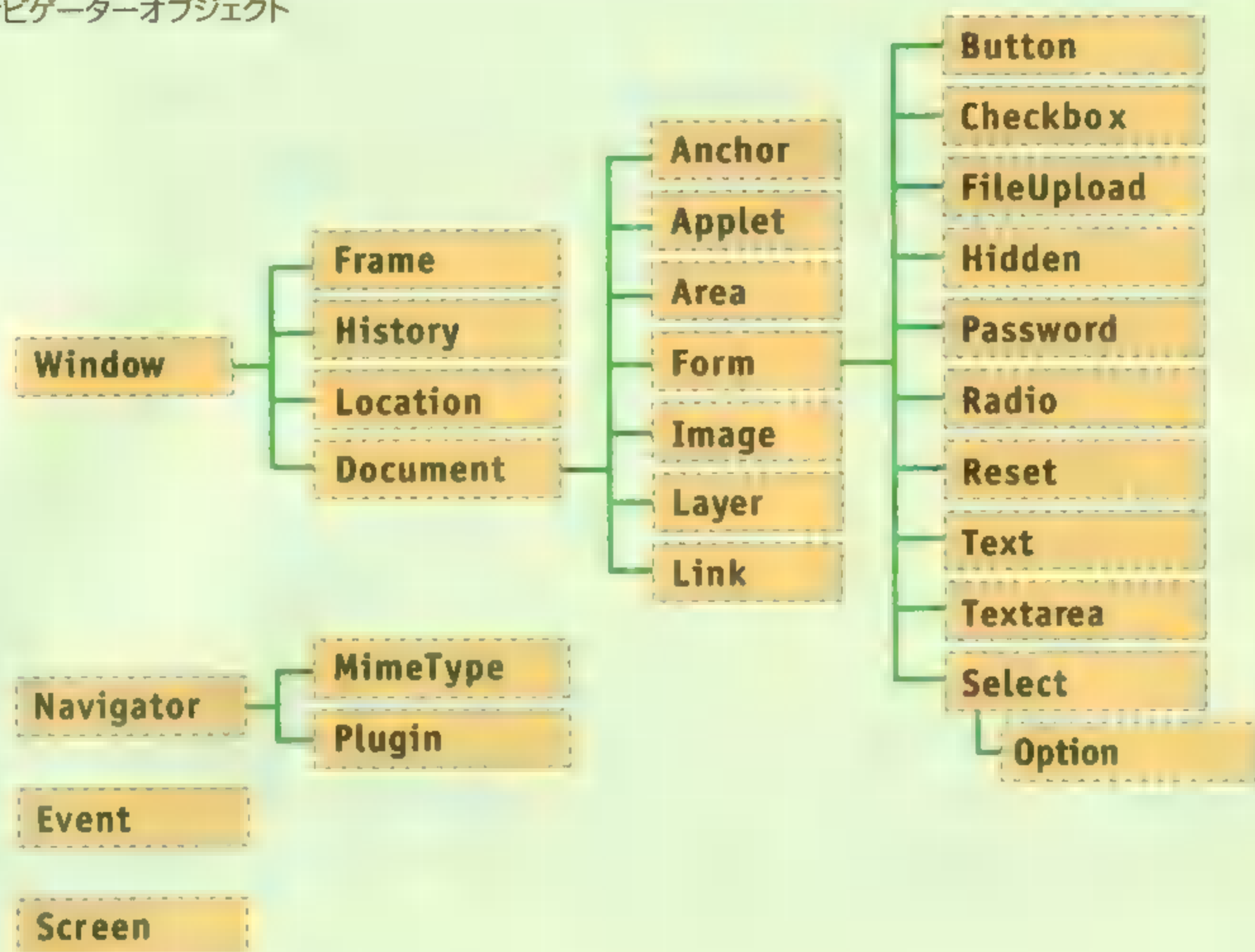
オブジェクトには大きく分けて、ビルトインオブジェクトとナビゲーターオブジェクトの2種類があります。

■ビルトインオブジェクト

ビルトインオブジェクトはJavaScriptにあらかじめ組み込まれた機能をオブジェクト化したもので、ECMAScriptで標準化された仕様に基づいています。配列、文字列、正規表現などを扱うことができます。

■ナビゲーターオブジェクト

ナビゲーターオブジェクトはWebブラウザ自体の情報や、ページの表示に関わる機能／部品をオブジェクト化したもので、ウィンドウ、フォーム、画像などを扱うことができます。ECMAScriptで標準化されていないため、ブラウザ独自のオブジェクトやメソッド、プロパティが実装されていることがあります。

ナビゲーターオブジェクトを構成するオブジェクトは、右図のように基本的にはWindowオブジェクトを最上位とするツリー[illegible]をとっています。下位のオブジェクトは上位のオブジェクトのプロパティで参照でき、下位のオブジェクトを操作したい場合には上位のオブジェクトから[illegible]に「.」(ピリオド)でつなげて記述します。ただし、Windowオブジェクトは最上位のオブジェクトのため、省略できます。

JavaScriptのオブジェクト階層

それぞれのオブジェクトには、そのオブジェクトの状態を参照／変更するためのプロパティやオブジェクトに対して命令を実行するためのメソッドが多数用意されています。

本書で紹介するJavaScriptのオブジェクトとそれに付随するプロパティ、メソッドについてはp.417以降の「オブジェクト一覧」を参照してください。

プロパティ

プロパティとは、オブジェクトの状態や属性のことです。プロパティは参照や設定が可能ですが、プロパティによっては参照のみで設定のできないものもあります。

プロパティを参照するには次のようにオブジェクト名とプロパティ名を「.」(ピリオド)でつなげて記述します。

```
オブジェクト名.プロパティ名;
```

逆に設定する場合には次のようにします。

```
オブジェクト名.プロパティ名 = 値;
```

たとえば、Document(ドキュメント)オブジェクトにはドキュメントの背景色や文字色、ドキュメントのURI、タイトル、ドメインなどの状態や属性を表すプロパティがあります。これらを参照または設定することによって、ページの背景色を変えたり別のページに移動させたりするなどの動作を実現できます。次の例では、DocumentオブジェクトのbgColorプロパティでページの背景色を赤にしています。

```
例:document.bgColor = "red";
```

メソッド

メソッドとはオブジェクトに対する処理をまとめたものです。メソッドを呼び出すと、そのオブジェクトに対して何らかの処理をさせることができます。

メソッドを記述するには次のようにオブジェクト名とメソッド名を「.」(ピリオド)でつなげて記述し、必要に応じて()内に引数を指定します。

```
オブジェクト名.メソッド名();
```

たとえば、Stringオブジェクトには文字列の色やサイズを変更したり、文字列の検索や分割を行ったりするメソッドがあります。次の例では、StringオブジェクトのindexOfメソッドで文字列strから"Internet"を検索し、結果を変数iに代入しています。

```
例:str = "Microsoft Internet Explorer";
   i = str.indexOf("Internet", 0);
```

BASIC.06

イベント

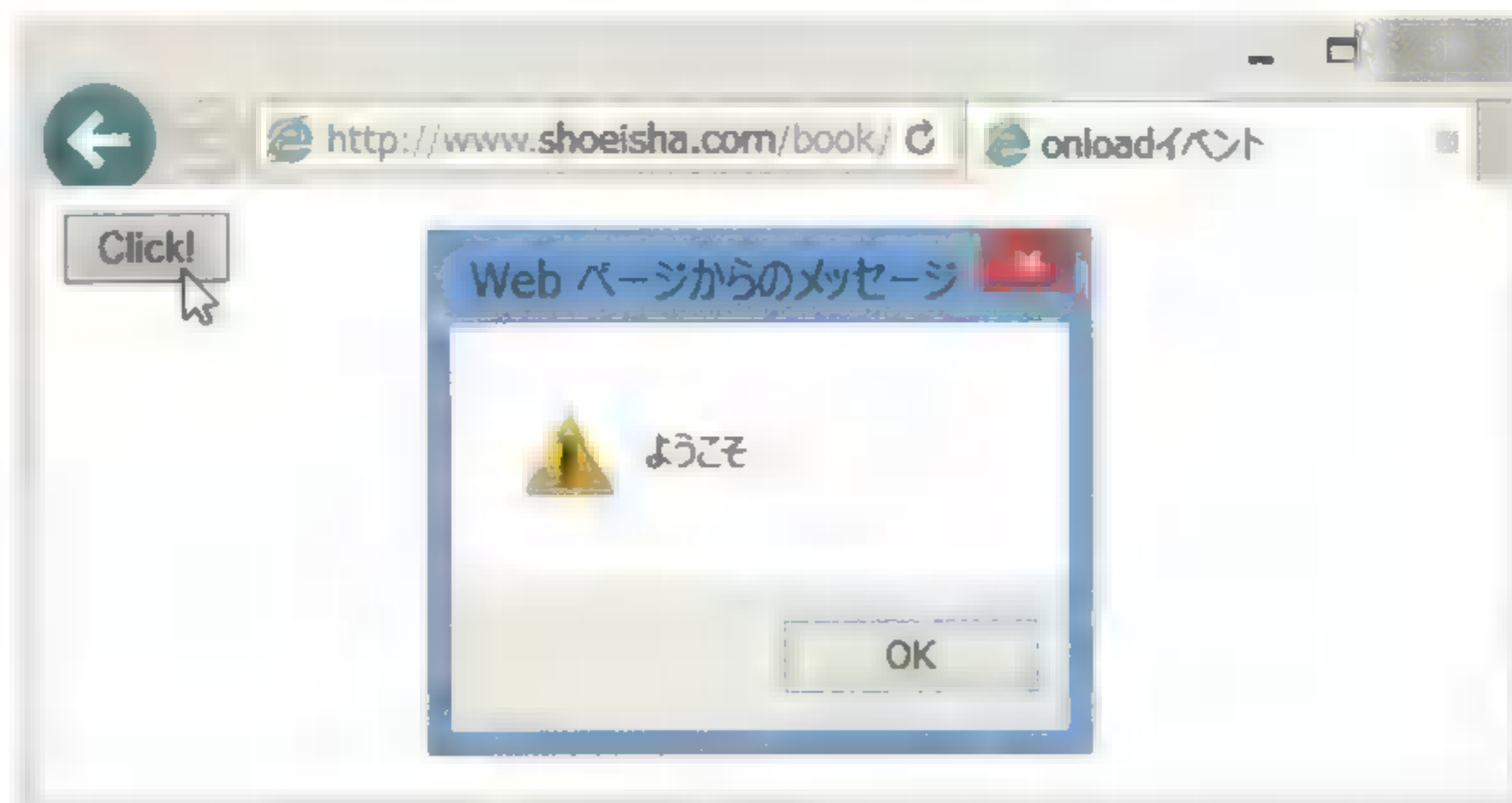
イベントはマウスのボタンがクリックされた、ページの読み込みが完了した、フォームの選択メニューが変更された、など何かしらの動作が起こったときに発生します。このイベントを取得するのがイベントハンドラです。たとえば、フォームのボタンなどがクリックされたときのイベントハンドラはonclickになります。

イベントでスクリプトを実行する

イベントハンドラをHTML/XHTMLの要素の属性(イベント属性)やオブジェクトに設定し、JavaScriptの処理を呼び出すようにすれば、指定の動作が起こったときに設定した処理を実行させることができます。その■、「;」(セミコロン)で区切れば複数の処理を指定できます。

■イベント属性を利用する

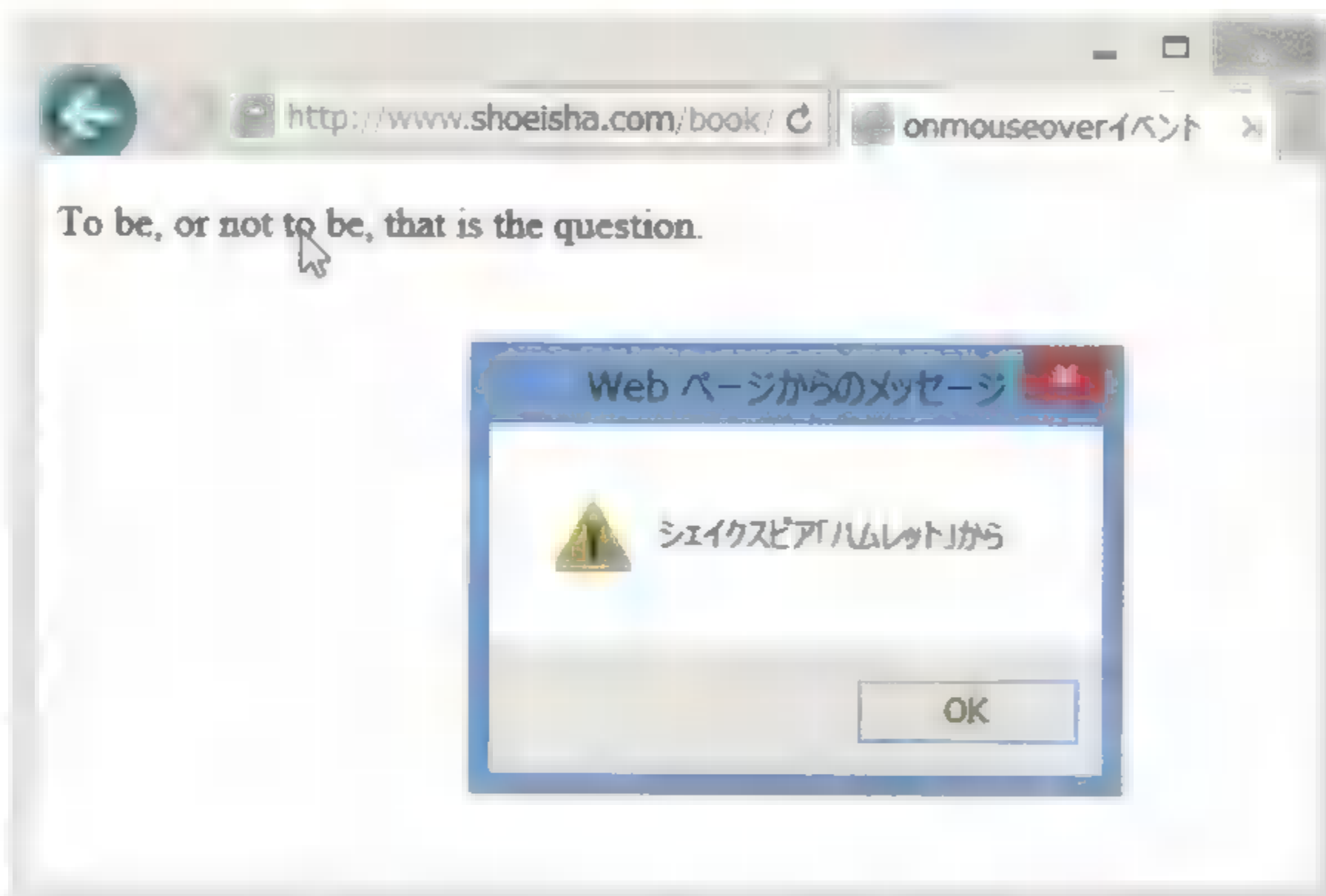
HTML/XHTML要素の属性を利用し、実行するJavaScriptの処理を「"" 」(ダブルクォーテーション)で囲って指定する方法です。

例:
```
<input type="button" value="Click!"
  onclick="alert('ようこそ');" />
```

クリックイベント(onclick)。ボタンをクリックするとalert('ようこそ')が実行されます

■スクリプト中に設定する

HTML/XHTMLのイベント属性を利用せず、JavaScriptでイベントハンドラの内容を直接指定する方法です。通常、関数の呼び出しには関数名の後ろに()が必要ですが(p.039)、この場合()は不要です。

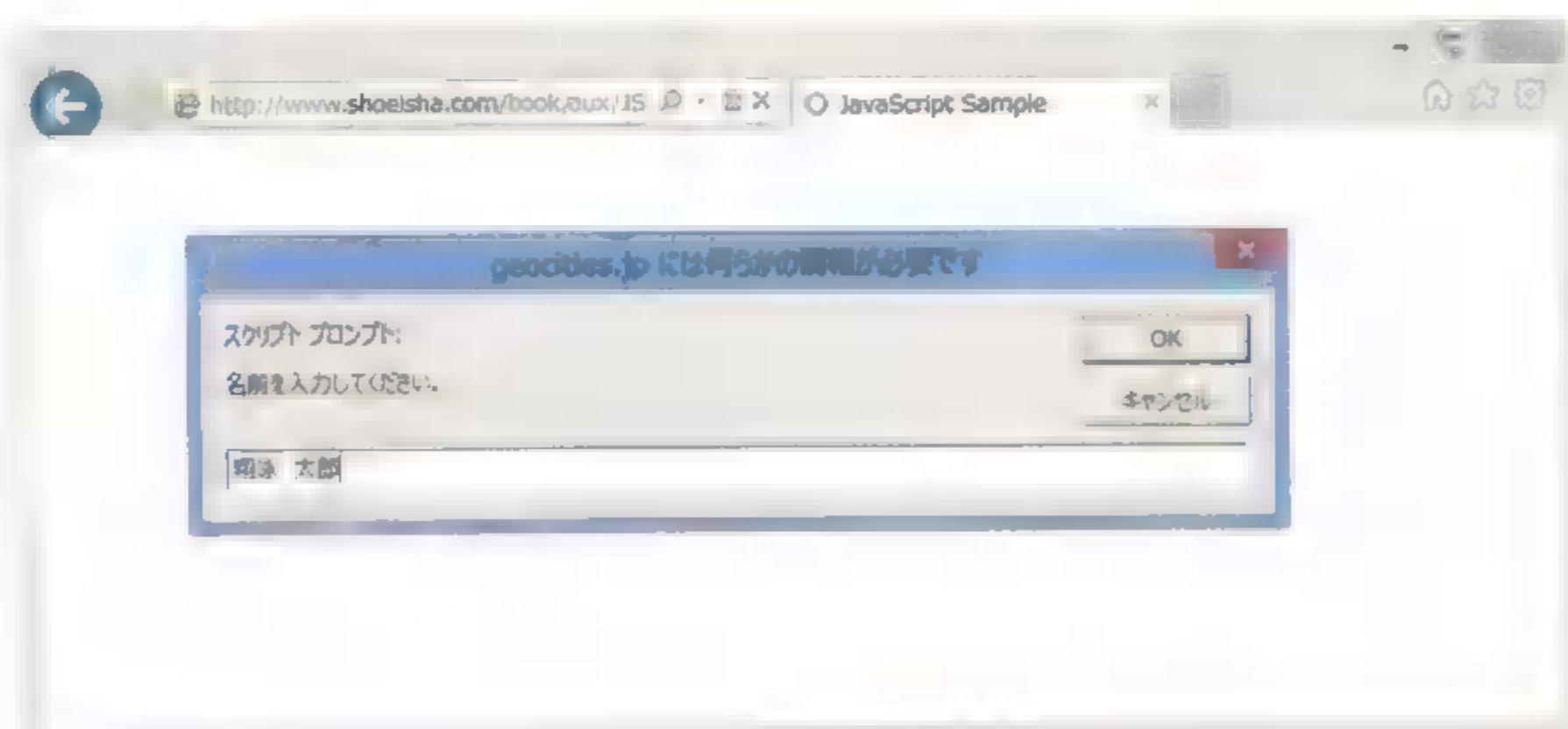
例：
```
function myFunc(){
        alert("シェイクスピア「ハムレット」から");
        return false;
}
document.getElementById("test").onmouseover = myFunc ;
```

BASIC.07

変数

変数とは値や式を格納する箱のようなものです。変数を用いることによってスクリプトが簡潔になり、メンテナンスが容易になります。

JavaScriptでは変数に数値、文字列、真偽値(trueまたはfalse)、オブジェクト(日付、配列、ウィンドウなど)を格納できます。

下記の例ではユーザーが入力した名前をresという変数に格納し、document.writeで書き出しています。

例：
```
var res = prompt("名前を入力してください。");
document.write("こんにちは、"+ res + "さん。");
```

実行結果。ユーザーが入力した名前を変数に格納し、書き出します

変数の記述方法

変数の記述方法にはいくつかのパターンがあります。変数は宣言してから利用するのが一般的です。変数の宣言は変数名の前に変数を宣言する命令文varを付けて行いますが、JavaScriptでは初めて登場する単語を変数として認識するため、このvarは省略可能です。ただし、varの有無によって変数の有効範囲が変わることがあるので注意が必要です(p.019を参照)。

■変数を宣言する

変数の宣言のみを行います。複数のときは例2のように1行にまとめることもできます。

```
var 変数名;
```

例1：
```
var str;
var myNum;
```

例2：
```
var str; myNum;
```

■変数を宣言し、初期値を代入する

変数を宣言すると同時に、初期値を代入します。複数のときは例2のように1行にまとめることもできます。

```
var 変数名 = 値;
```

例1：
```
var str = "ようこそ";
var myNum = 10;
```

例2：
```
var str = "ようこそ"; myNum = 10;
```

変数名

変数は、p.008のような命名規則に沿っていれば任意の名前を指定できます。p.017の例では*res*ではなく*henji*としても関係ありません。

変数名には、その変数にどんな種類の値が代入されているのかがわかりやすい名前を付けておくとよいでしょう。

なお、本書では変数名は *res* のように斜体で示してあります。

変数の有効範囲

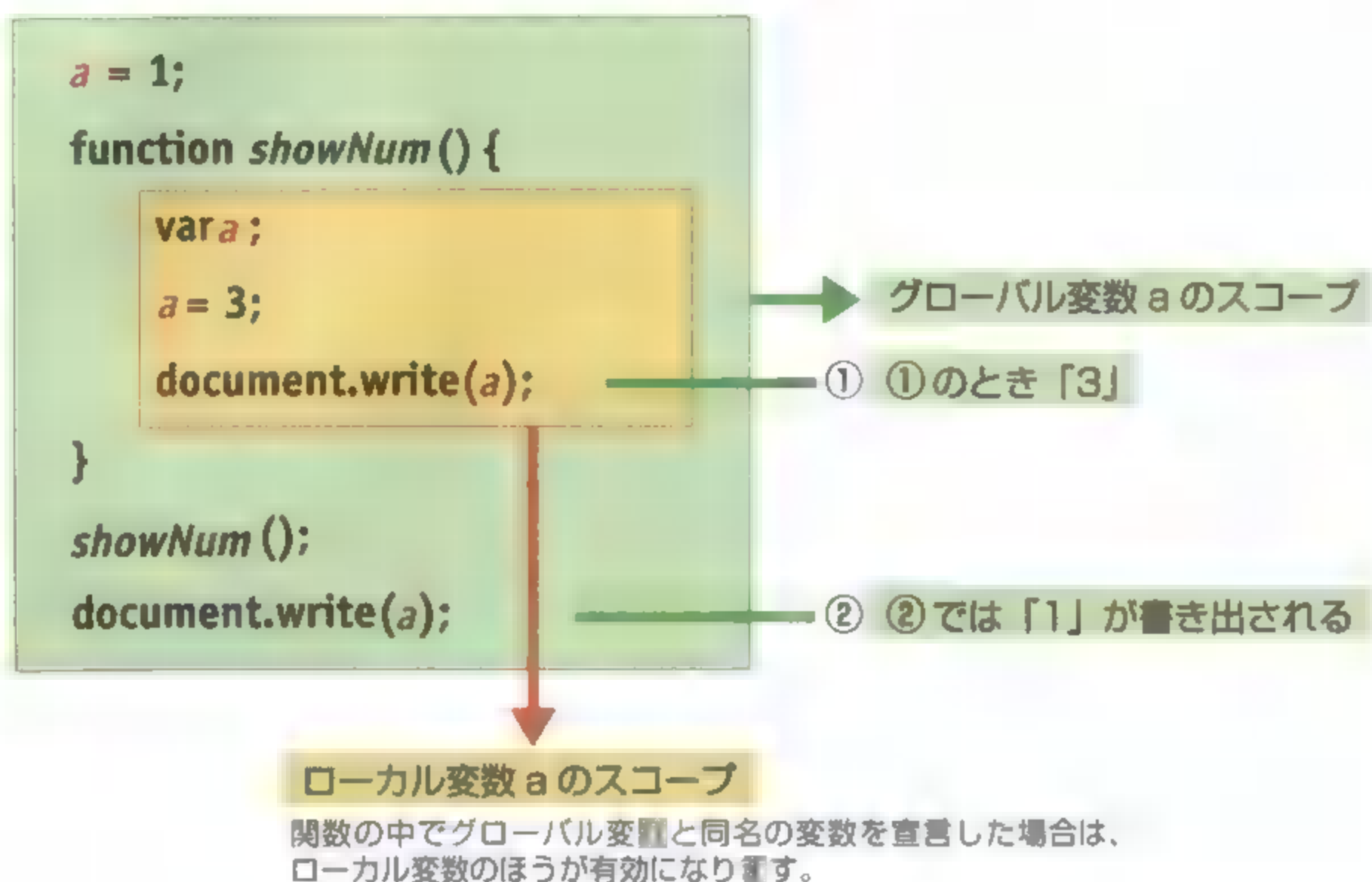
変数には何処でも有効なグローバル変数と、特定の関数(p.039)の内部でのみ有効なローカル変数があり、それぞれの変数が有効な範囲をスコープと言います。一般的なプログラム言語

とは違い、ブロック(p.025)はスコープにはなりません。

変数を宣言するとき、関数の内部でvarを付けて宣言した場合にはローカル変数となり、関数の外部で宣言した場合や関数の内部であってもvarを使わずに宣言した場合にはグローバル変数となります。混乱を避けるためにもなるべくvarを使って宣言をした方がよいでしょう。

関数の途中でvarを使って宣言しても、関数の冒頭で宣言したのと同じ扱いになることには注意が必要です(宣言と同時に代入した場合、代入は実際に記述された位置で行われます)。

関数の中でグローバル変数と同名の変数を宣言した場合は、ローカル変数のほうが有効になります。

BASIC.08

演算子

JavaScriptには計算や比較に利用するさまざまな演算子があります。大きく分けて、四則演算を行う算術演算子、数値をビットとして扱うビット演算子、if構文などで複数の条件を扱う際に使用する論理演算子、数値の大小を比較する比較演算子、演算の結果を変数に代入する代入演算子です。それぞれについて、例を挙げながら簡単に説明します。各演算子の詳細については、p.022の一覧表を参照してください。

算術演算子

加算、減算、乗算、除算、および除算の余りを求める演算子です。数値の計算に使用します。なお、日常的な計算においては、12+5=17のように使用しますが、JavaScriptでは多くの場合、代入演算子を使用して、次のように変数に計算の結果を代入します。

例：**a = 12 + 5;** //12+5の結果を変数aに代入する(=は代入演算子)

ビット演算子

ビット演算子では式の数値を32ビットの整数と見なして演算を行います。たとえば、次のようになります。

例1：**12 & 5 は 4(1100 & 0101 = 0100)**
例2：**12 | 5 は 13(1100 | 0101 = 1101)**

HTML/XHTMLと組み合わせて利用する限り、ビット演算子が登場するケースはあまりありません。

論理演算子

「両方が正しい」、「どちらか一方が正しい」、「正しくない」など、条件を判別するための演算子です。主にif構文(p.024参照)の条件式に使用します。また、多くの場合は比較演算子と組み合わせて次のように使用します。

```
例：if(a >= 5 && b >= 10){
        // aが5以上で、bが10以上の場合の処理
    }
```

比較演算子

2つの値の比較を行う演算子です。数値の大小を比較したり、文字列の一致を調べたり、値がtrueとfalseのどちらであるかを調べたりする際に使用します。結果はtrueまたはfalseのいずれかになり、さまざまな構文の条件式に使用されます。以下にif構文(p.024参照)での例を示します。

```
例1：if(a < 10){
        // aが10未満の場合の処理
     }

例2：if(name != ""){
        // 変数nameが空白文字ではない場合の処理
     }

例3：if(res == true){
        // 変数resの値がtrueである場合の処理
     }
```

代入演算子(複合代入)

JavaScriptにおける「=」(イコール)は左辺の変数に右辺の値を代入するためのものです。たとえば、演算の結果や文字列の結合結果を代入します。

```
例1：a = 12 + 5; //演算の結果を代入
例2：name = "田村" + "明"; //文字列の結合を代入
```

また、代入演算子「+=」と組み合わせて次のように使用することもできます。

```
例：name = "田村" + "明";
    name += "さん" // nameの値は"田村明さん"となる
```

演算子の一覧

■演算子一覧

算術演算子				
+	加算	2つの数値の和を求める	a+b	aにbを加える
-	減算	2つの数値の差を求める	a-b	aからbを引く
*	乗算	2つの数値の積を求める	a*b	aとbを掛ける
/	除算	2つの数値で除算を行う	a/b	aをbで割る
%	余剰	2つの数値で除算し、その余りを求める	a%b	aをbで割った余り
++	インクリメント	変数の値を1増やす	a++	aの値を1増やす(a=a+1と同じ)
--	デクリメント	変数の値を1減らす	a--	aの値を1減らす(a=a-1と同じ)

<table>
<tr><th colspan="3">ビット演算子</th></tr>
<tr><td>&</td><td>ビットごとのAND</td><td>2つの値で各ビットの論理積を求める</td></tr>
<tr><td>|</td><td>ビットごとのOR</td><td>2つの値で各ビットの論理和を求める</td></tr>
<tr><td>^</td><td>ビットごとのXOR</td><td>2つの値で各ビットの排他的論理和を求める</td></tr>
<tr><td>~</td><td>ビットごとのNOT</td><td>各ビットを反転させた結果を求める</td></tr>
<tr><td><<</td><td>左シフト</td><td>指定された分だけ各ビットを左にシフトする</td></tr>
<tr><td>>></td><td>右シフト</td><td>指定された分だけ各ビットを右にシフトする</td></tr>
<tr><td>>>></td><td>符号なし右シフト</td><td>符合を考慮せず、右シフトする</td></tr>
</table>

<table>
<tr><th colspan="3">論理演算子</th></tr>
<tr><td>&&</td><td>論理積</td><td>2つの値の論理積を求める(両方がtrueの場合のみtrue)</td></tr>
<tr><td>||</td><td>論理和</td><td>2つの値の論理和を求める(一方または両方がtrueならtrue)</td></tr>
<tr><td>!</td><td>論理否定</td><td>指定された値の論理否定を求める</td></tr>
<tr><td>? :</td><td>条件</td><td>?の前の条件がtrueなら:の左側の文、falseなら:の右側の文を実行する</td></tr>
<tr><td>,</td><td>カンマ</td><td>2つの式を順に続けて実行する</td></tr>
</table>

比較演算子			
<	より小さい	a<10	aが10未満のときtrue、10以上のときfalse
<=	以下	a<=10	aが10以下のときtrue、10より大きいときfalse
>	より大きい	a>10	aが11以上のときtrue、10以下のときfalse
>=	以上	a>=10	aが10以上のときtrue、10未満のときfalse
==	等しい	a==10	aが10のときtrue、それ以外のときfalse
!=	等しくない	a!=10	aが10以外のときtrue、10のときfalse

代入演算子(複合代入) *算術演算子だけでなくビット演算子についても=と組み合わせて同様に記述できます				
=	代入	値を変数に代入する	a=5	aに5を代入する
+=	加算	指定された値を加算する	a+=5	aに5を加える(a=a+5と同じ)
-=	減算	指定された値を減算する	a-=5	aから5を引く(a=a-5と同じ)
*=	乗算	指定された値を乗算する	a*=5	aに5を掛ける(a=a*5と同じ)
/=	除算	指定された値で除算する	a/=5	aを5で割る(a=a/5と同じ)
%=	余剰	指定された値で除算し、余りを求める	a%=5	aを5で割った余り(a=a%5と同じ)

演算子の優先順位

演算子には優先順位があり、演算はその順序に従って行われます。式に同じ優先順位の演算子があった場合は、■本的には左側から■番に演算が行われます。

下の例では演算子の優先順位に従って演算がなされ、結果は42になります。[10*4=40, 6/3=2, 40+2=42]

例：**a = 10*4+6/3;**

命令を( )で囲むと、演算子の優先順位に■わらず括弧内の計算が先に行われ、下の例の結果は60となります。 [6/3=2, 4+2=6, 10*6=60]

例：**b = 10*(4+6/3);**

下の表は演算子の優先順位を高い順に並べたものです。同じ行にあるものの優先順位は同等になります。

優先順位	演算子
1	. [] ()
2	++ -- - ~ !
3	* / %
4	+ -
5	<< >> >>>
6	< <= > >=
7	== != === !==
8	&
9	^
10	|
11	&&
12	||
13	?:
14	= += -= *= /= %= &= |= ^= <<= >>= >>>=

BASIC.09

条件分岐

プログラムは通常上から下へ順番に処理されていきますが、複雑なプログラムではユーザーの動作や環境などによって処理を分ける必要が生じます。条件によって処理を分ける条件分岐の構文には、2通りの処理に分岐する「if～else構文」と、複数の処理に分岐する「switch～case構文」があります。

条件が真か偽かで処理を分ける（if～else構文）

★……条件
◆……条件★が真である場合の処理
▲……条件★が偽である場合の処理

if文は、条件分岐の制御構文の1つです。

条件文★がtrue（真）の場合は処理◆が実行され、false（偽）の場合は処理▲が実行されます。trueの場合のみ処理を行いたい場合は、else以下を記述しません。else if……のようにelseの後にifを続けて、さらに条件を分岐させることも可能です。

記述方法	処理の流れ
if(条件a) 処理1	条件a → true → 処理1

if(条件a)
処理1
else
処理2
	条件a → true → 処理1
	条件a → false → 処理2

if(条件a)
処理1
else if(条件b)
処理2
else
処理3
	条件a → true → 処理1
	条件a → false → 条件b
	条件b → true → 処理2
	条件b → false → 処理3

なお、複数の処理を行いたい場合には、それらの処理全体を{}でくくります（下記コラム参照）。処理が1行だけであれば、if(★) ◆;のように{}を省略できます。

Column　複文（ブロック）

{}で囲まれた部分は複文（ブロック）になります。
if構文やwhile構文などでは、すぐ後ろの1つの命令しか実行されませんが、{}で囲むことによって、複数の処理をまとめることができます。

```
if( 条件1 )
処理1; //条件1のとき実行されるのは処理1のみ
処理2;

if( 条件1 ){
処理1; //条件1のとき実行されるのは処理1と処理2
処理2;
}
```

このように{ } で囲んだ場合は1つのブロックと見なされるので、その中に記述されているすべての処理が実行されます。

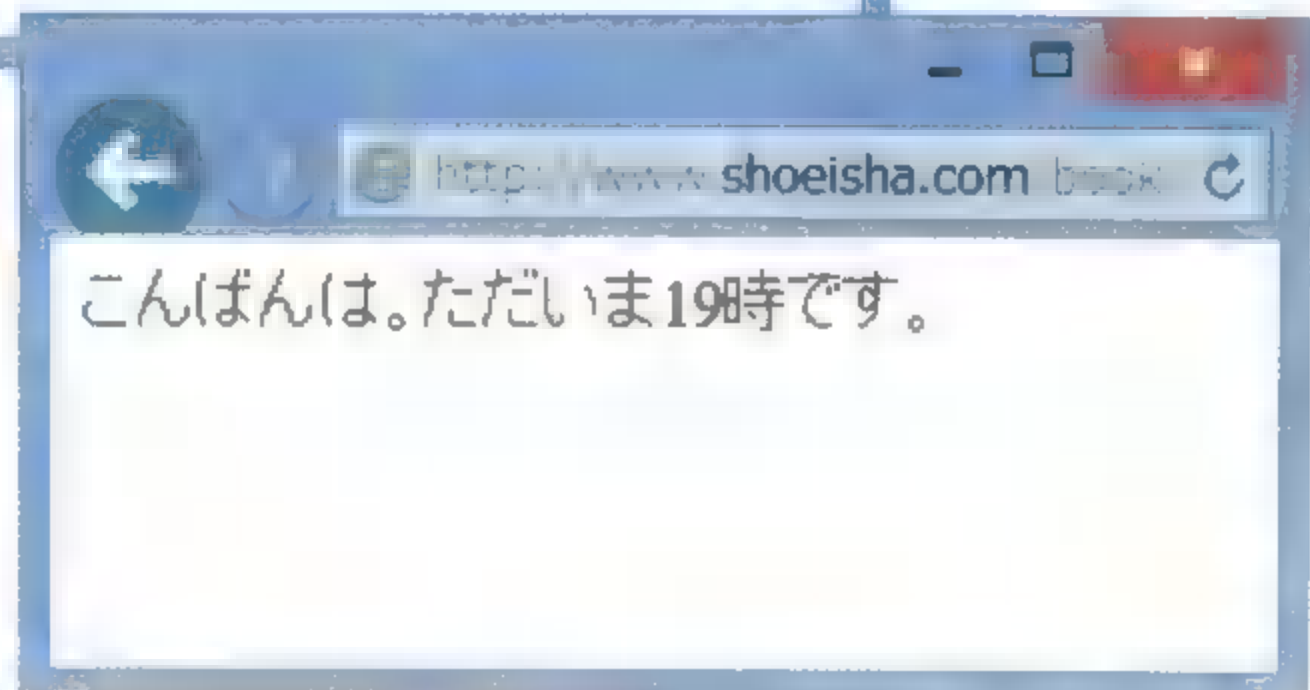
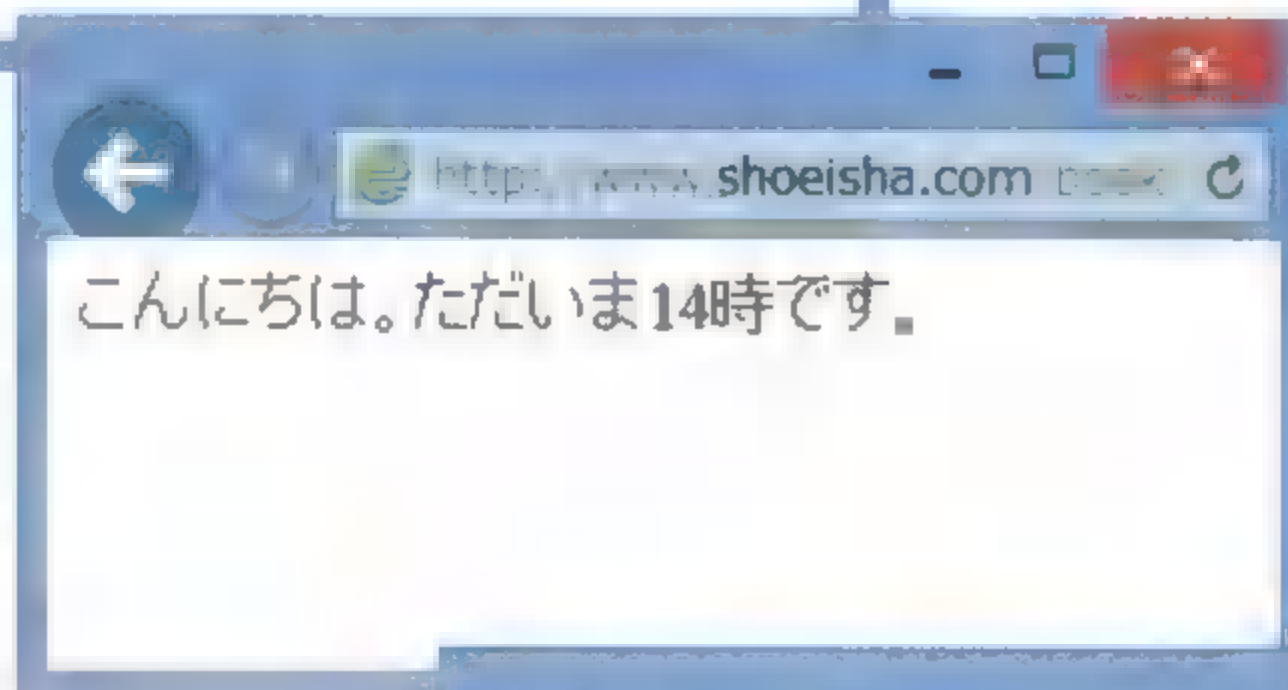
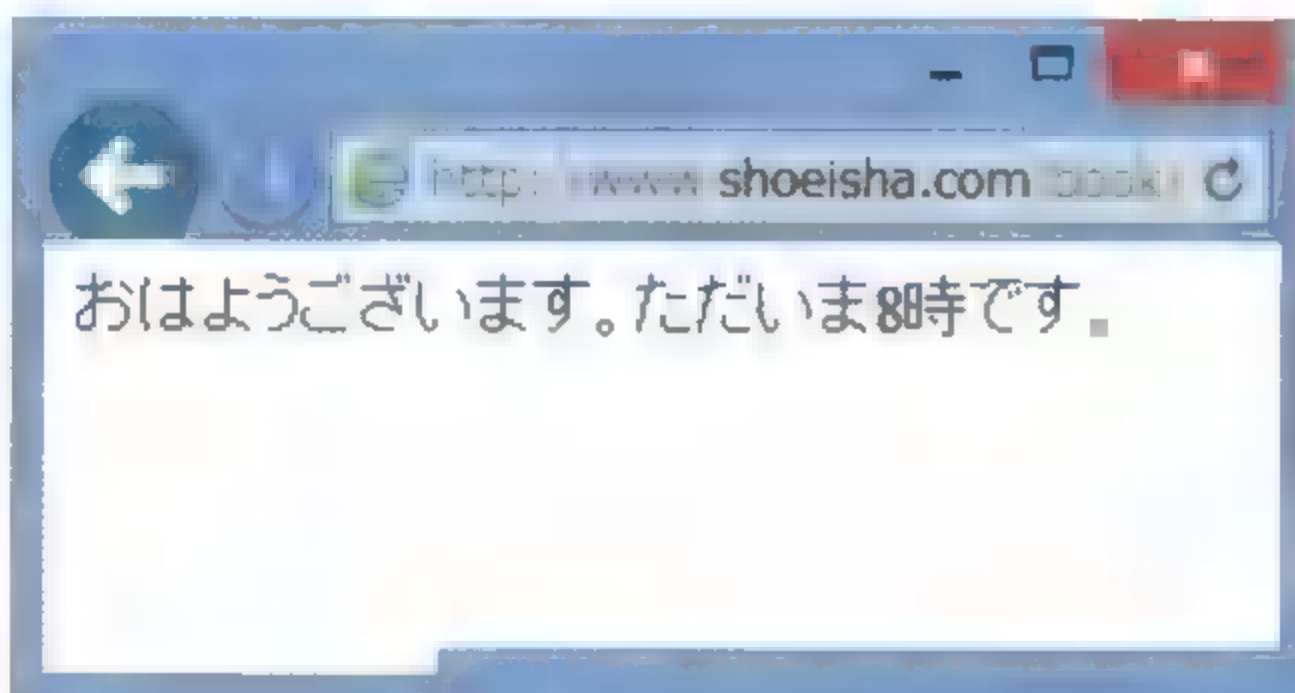
アクセスされた時刻を取得して、その時刻の範囲に応じてif構文で処理を分けています。

Source

```
today = new Date();
h = today.getHours();
if((h > 5) && (h <= 10)) // 条件1:6時から10時までの場合
    document.write("おはようございます。");
else if((h > 10) && (h <= 17)) // 条件1が偽でかつ、条件2:11時から17時までの場合
    document.write("こんにちは。");
else // 条件2も偽:それ以外の場合
    document.write("こんばんは。");
document.write("ただいま"+ h + "時です。"); //すべてで実行される処理
```

おはようございます。ただいま8時です。

こんにちは。ただいま14時です。

こんばんは。ただいま19時です。

アクセスされた時刻を取得して、その時刻の範囲に応じて処理を分けています。

条件によって複数の処理を振り分ける（switch~case構文）

```
switch(★) {
    case ◆1: ▲1; break;
    case ◆2: ▲2; break; ...
    default: ●;
}
```

★……条件
◆……値（◆1, ◆2, ...）
▲……処理（▲1, ▲2, ...）
●……処理

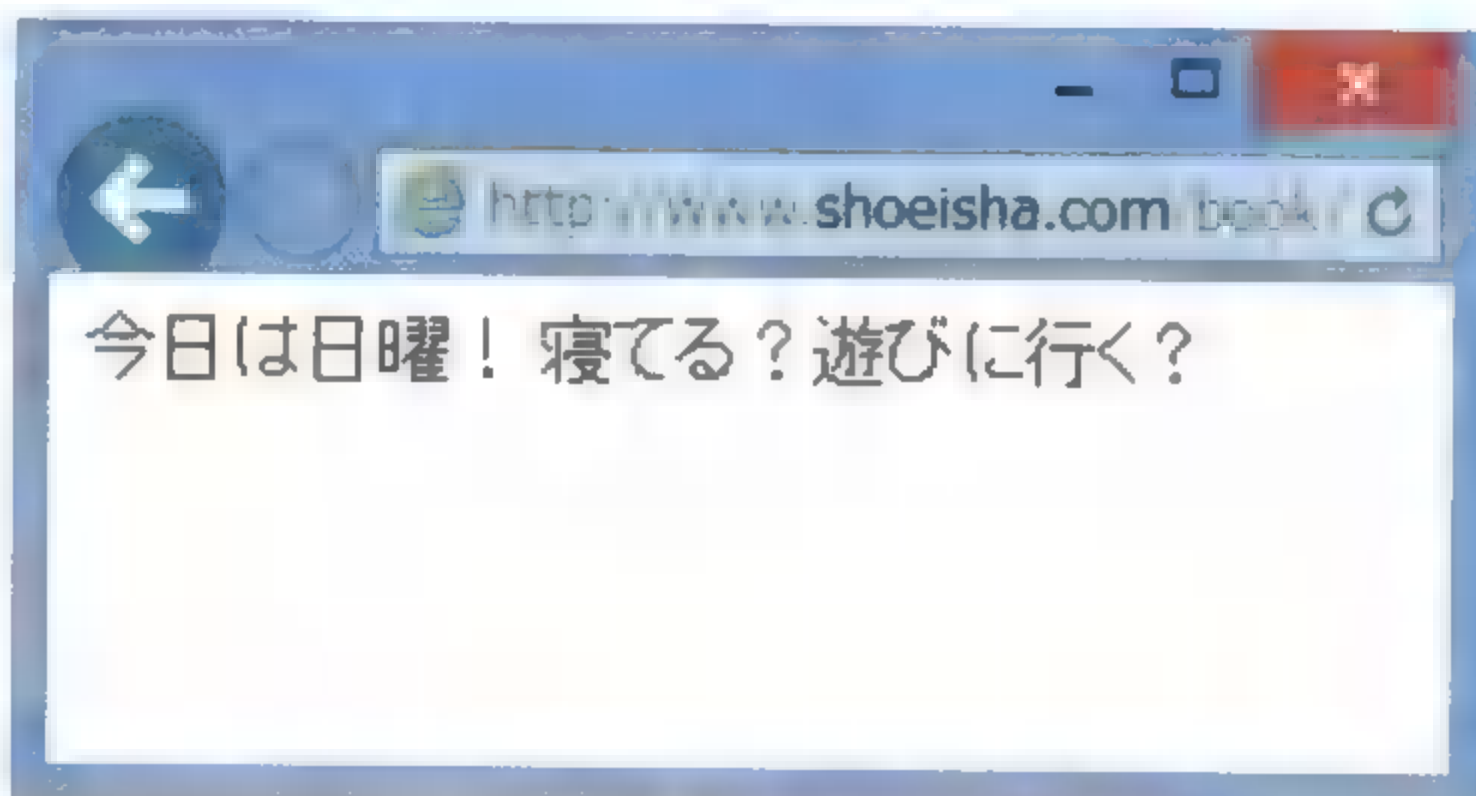
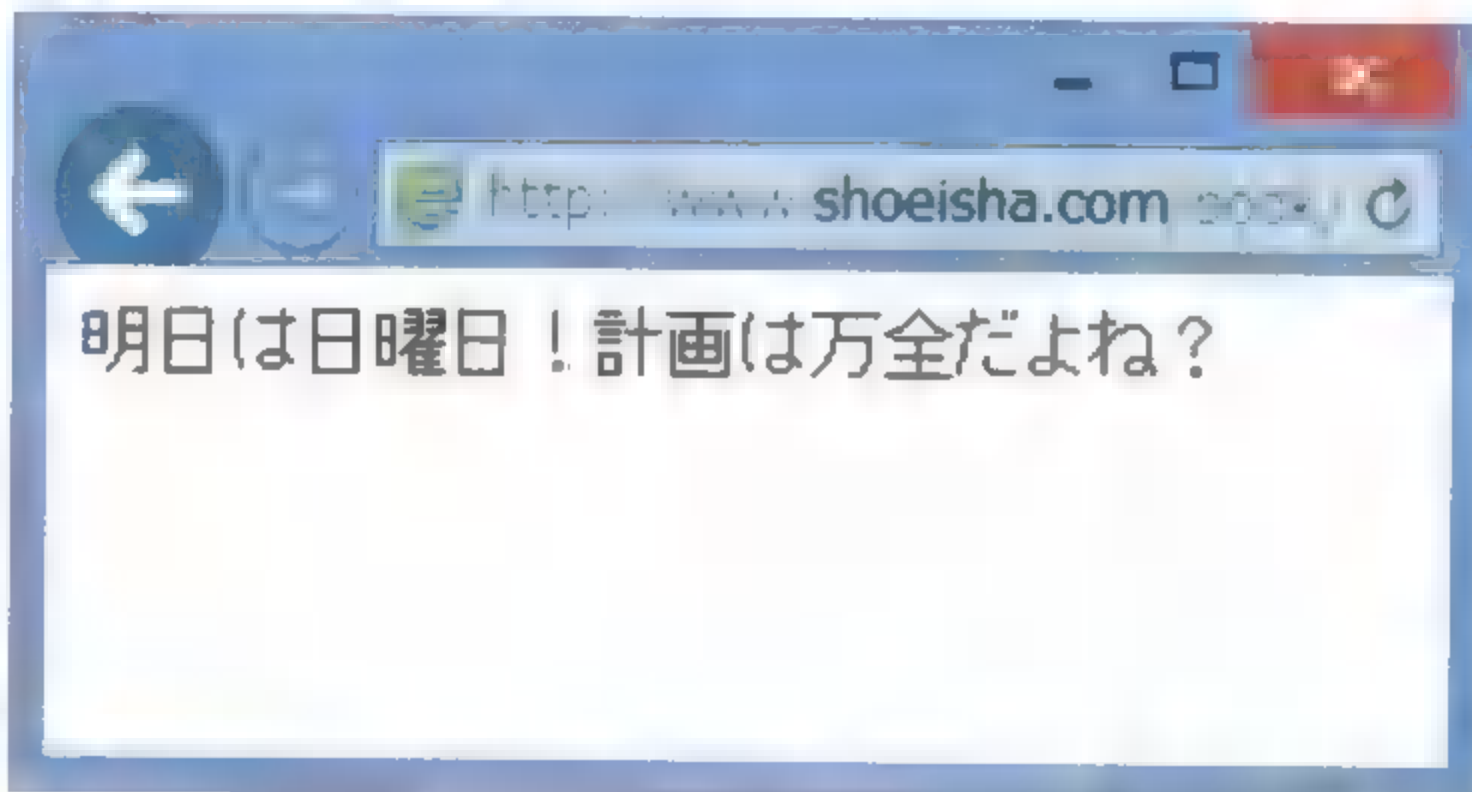
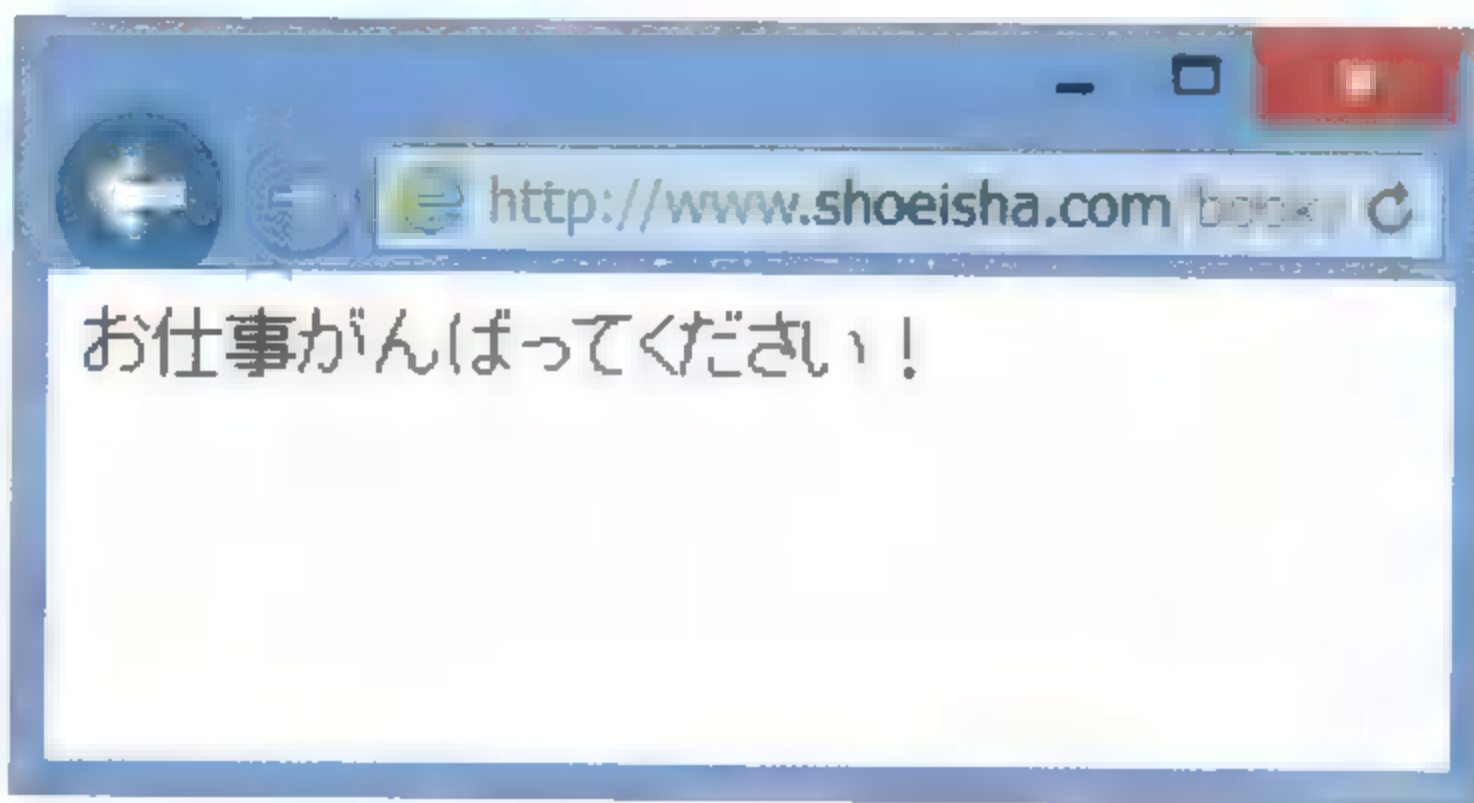
switch文はcaseで定義された複数の選択肢の中から条件の値に合うものを選び、その処理を実行します。条件の値がcaseで定義したどの値にも当てはまらなかった場合は、defaultに進みます。

処理を抜けるにはbreak文を記述します（p.035参照）。たとえば次の例では、★が◆1のとき▲1のみが処理され、★が◆2のとき▲2のみが処理されます。

```
例：switch(★) {
        case ◆1: ▲1;
        case ◆2: ▲2; break;
        case ◆3: ▲3; break; ...
    }
```

サンプルでは曜日を0から6までの数値（日曜=0、月曜=1……土曜=6）で返すメソッドgetDay()の値によって、swich構文で処理を分岐しています。

Source

```
today = new Date();
switch(today.getDay()){
    case 0: // 値0(日曜)を返す場合
        document.write("今日は日曜！寝てる？遊びに行く？");
        break;
    case 6: // 値6(土曜)を返す場合
        document.write("明日は日曜日！計画は万全だよね？");
        break;
    default: // その他の値(平日)の場合
        document.write("お仕事がんばってください！");
}
```

アクセスした曜日によって表示されるメッセージが変化します

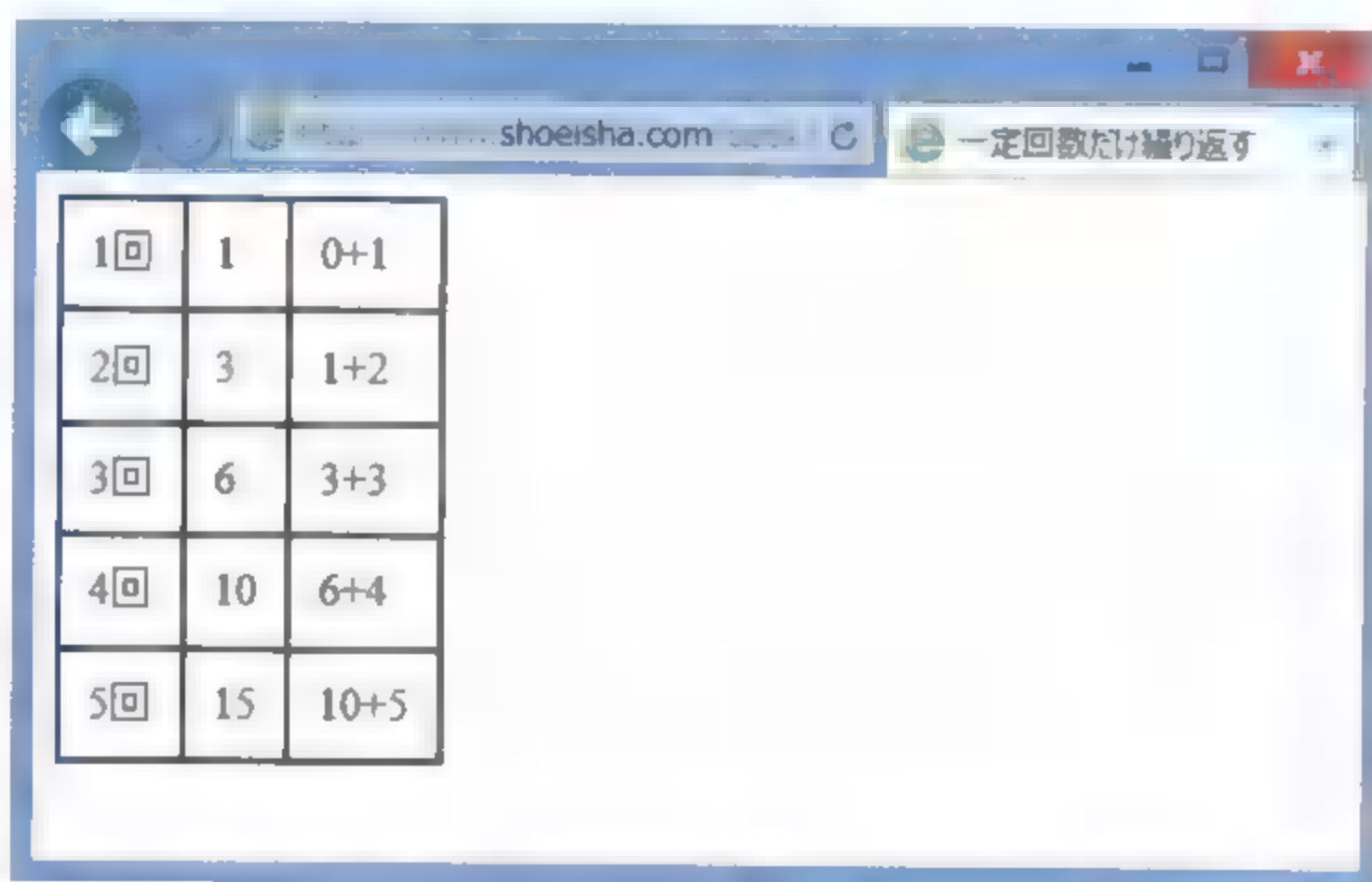
BASIC.10

繰り返し処理

処理を繰り返す構文には、繰り返す回数を指定するfor文と、条件が満たされている間繰り返すwhile文があります。処理を繰り返す回数が決まっているときはfor文、繰り返す回数が決まっていないときはwhile文を使用します。

指定回数だけ処理を繰り返す（for文）

```
for(★; ◆; ▲){
      ●
}
```

★……初期値
◆……条件
▲……変数の増減
●……処理

処理を指定回繰り返したいときに使用する構文です。まず変数に初期値★を設定し、条件◆がtrueの間、処理●を実行します。処理を1回実行するごとに、▲にある式に基づき変数の値が増減します。

1から5までの数を加算していくサンプルです。変数iの初期値を1とし、iが6より小さい間(5以下の間)、for文で処理を繰り返します。処理を1回行うごとにiは1増加(i++)していきますので、処理は5回行われることになります。

Source

```
total = 0;
for(i=1; i<6; i++){ …処理を5回繰り返す
    document.write("<tr><td>"+ i+ "回</td>"); …回数を書き出す
    total = total + i ;
    document.write("<td>"+ total+ "</td>"); …計算結果を書き出す
    document.write("<td>"+ total-i+ "+", i+ "</td></tr>");
}
```

1回	1	0+1
2回	3	1+2
3回	6	3+3
4回	10	6+4
5回	15	10+5

1から5までの数を1ずつ加算していきます

すべての値に処理を繰り返す（for文）

```
for(★ in ◆){
    ▲
}
```

★……変数名
◆……オブジェクト名または配列名
▲……処理

オブジェクト◆の持つ値または配列の◆すべてに対して、それぞれ順番に▲を実行します。変数★にはオブジェクトや配列のインデックスが順番に格納されます。

「{属性の名前:値, 属性の名前:値, ...}」という書式でそれぞれの属性がそれぞれの値を持つオブジェクトを作成できます。ここでは作成したオブジェクトの属性をすべて書き出しています。配列では属性の名前の代わりにインデックスの数字が列挙されます。

Source ※レイアウトは外部CSSで指定しています

```
var obj = { name : "ジョン", age : "27",  email : "test@example.com", tel : "XX-1234-5678" };
var output = document.getElementById("output");
for(i in obj){ //オブジェクト全体に処理を行う
    output.innerHTML += "<tr><td>"+ i + "</td><td>"+ obj[i] + "</td></tr>";
}
```

作成したオブジェクトで参照できるすべてのプロパティとその値が書き出されます

条件が真の間、処理を繰り返す①（While文）

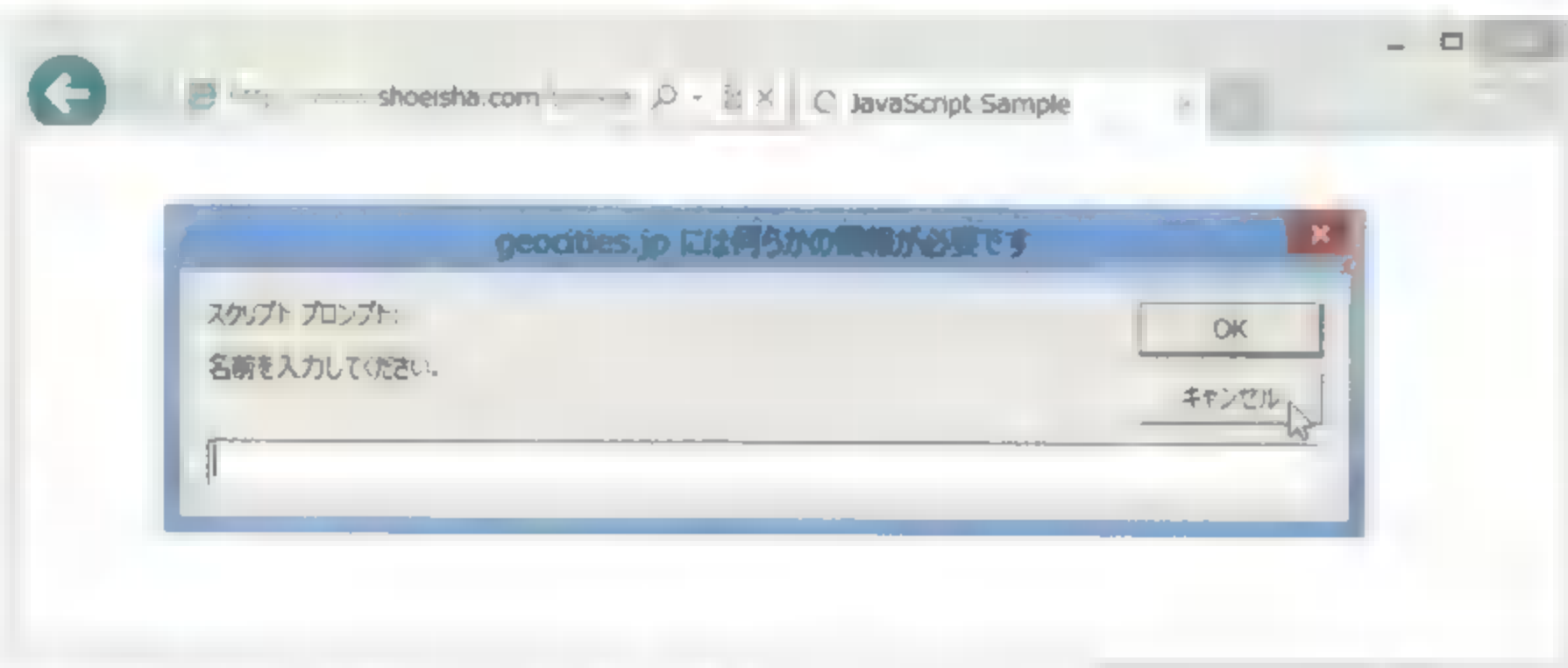
```
while(★){
      ◆
}
```

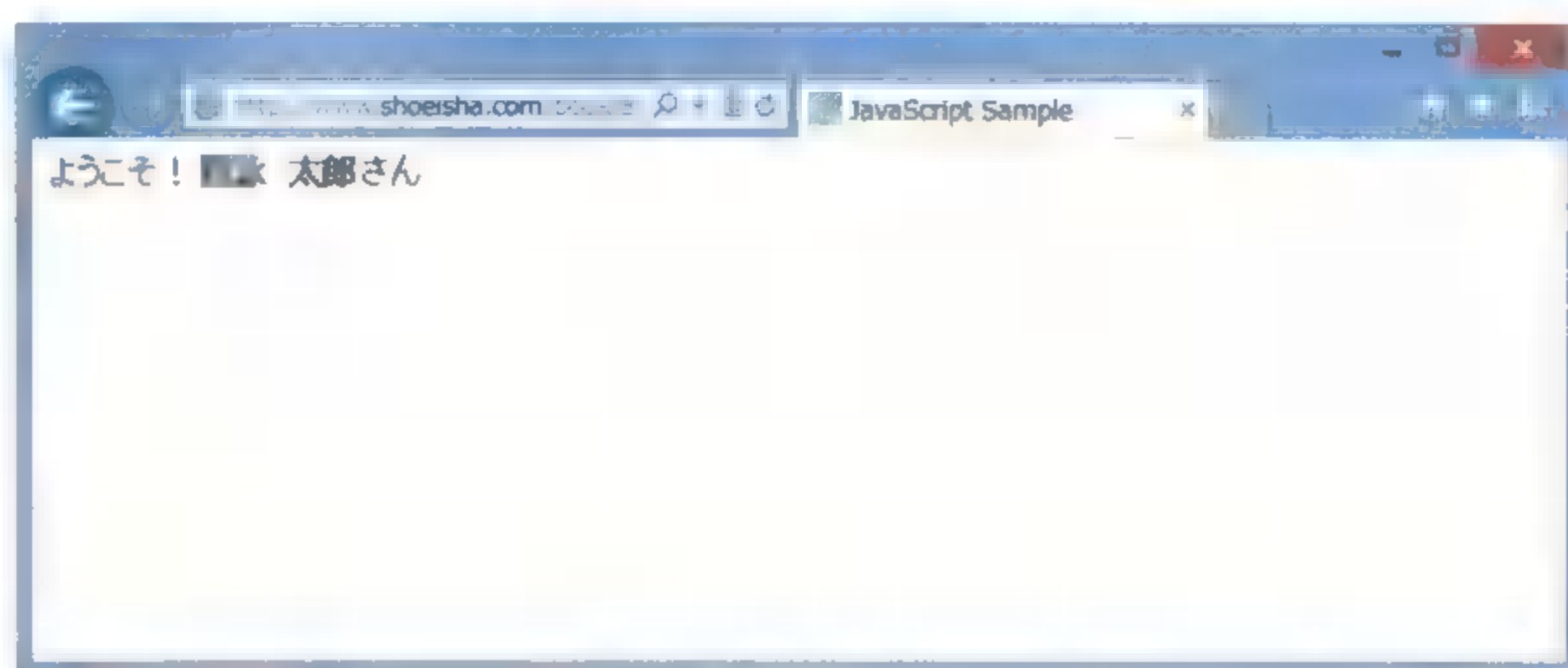
★……条件
◆……処理

条件が満たされるまで、処理を繰り返す制御構文です。
条件★がtrue（真）の間、処理◆を繰り返します。最初から条件が満たされていなかった場合、処理◆は一度も行われません。

入力された名前を表示するサンプルです。サンプルでは、まず変数resの初期値を空の文字列("")に設定しておきます。その後、文字入力ダイアログを表示して名前の入力を促し、[キャンセル]ボタンがクリックされた場合と何も入力されていない場合はwhile文で処理を繰り返しています。

Source

```
res = ""; //変数に空の文字列を設定
while((res == "") || (res == null)) //入力欄が空欄かキャンセルの場合、処理を繰り返します
    res = prompt("名前を入力してください。","");
document.write("ようこそ！ <b>"+ res + "</b> さん");
```

[キャンセル]ボタンをクリック。または空欄のまま[OK]ボタンをクリックすると、ダイアログが繰り返し表示されます

名前を入力し、[OK]ボタンをクリックすると、変数resに値が代入されます

繰り返しが終了し、先の処理が行われます

条件が真の間、処理を繰り返す②（While文）

```
do {
    ◆
} while(★)
```

★……条件
◆……処理

条件★がtrue（真）の間、処理◆を繰り返します。前項のwhile文では最初から条件が満たされていた場合は一度も処理が行われませんが、「do～while文」では処理の実行後に条件判断が行われるため、最初から条件が満たされていなかった場合でも一度は処理が実行されます。

まず文字入力ダイアログを表示して名前の入力を促し、[キャンセル]ボタンがクリックされた場合と何も入力されていない場合はdo ～ while文で処理を繰り返しています。
実行結果は前項のサンプルと同様です。

Source

```
do {
    res = prompt("名前を入力してください。","");
} while((res == "") || (res == null));
document.write("ようこそ！ <b>"+ res + "</b> さん");
```

BASIC.11

繰り返しの制御

break　処理から抜け出す
continue　繰り返し処理の先頭に戻る

繰り返し処理や分岐処理から抜け出す構文です。

■break

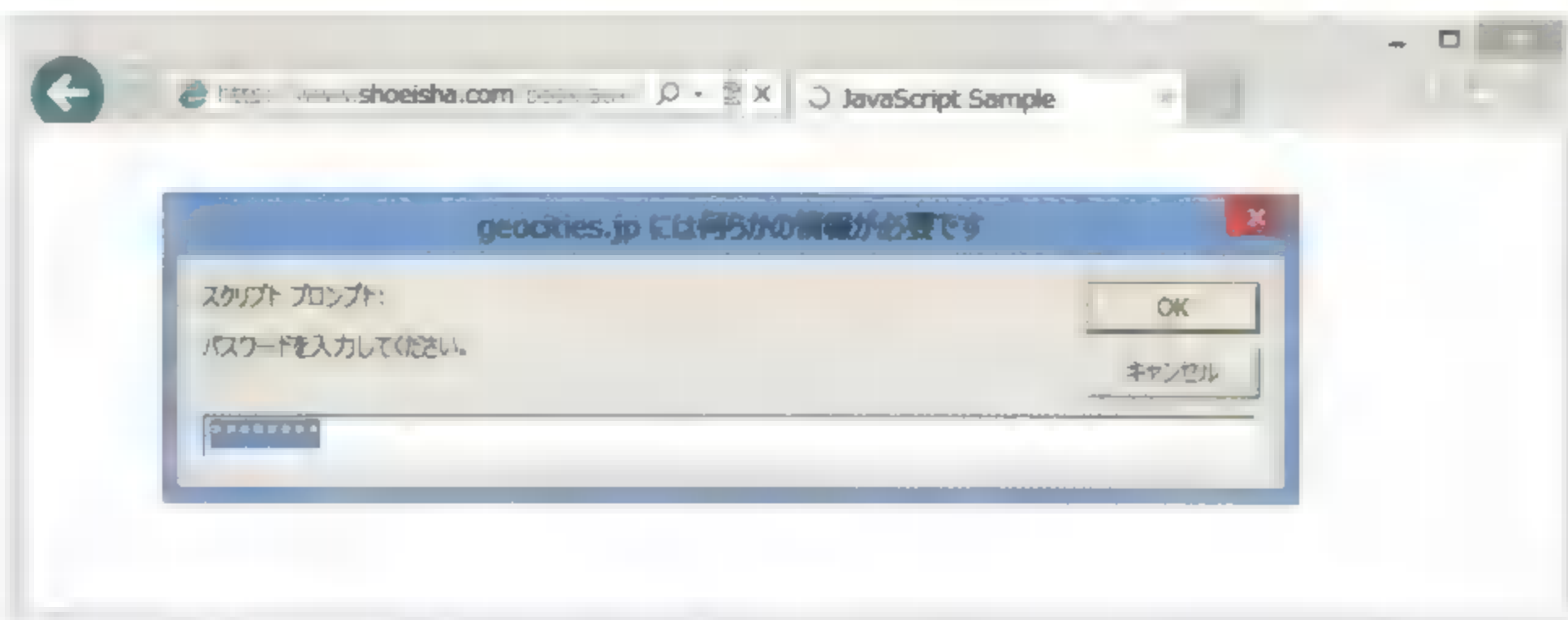
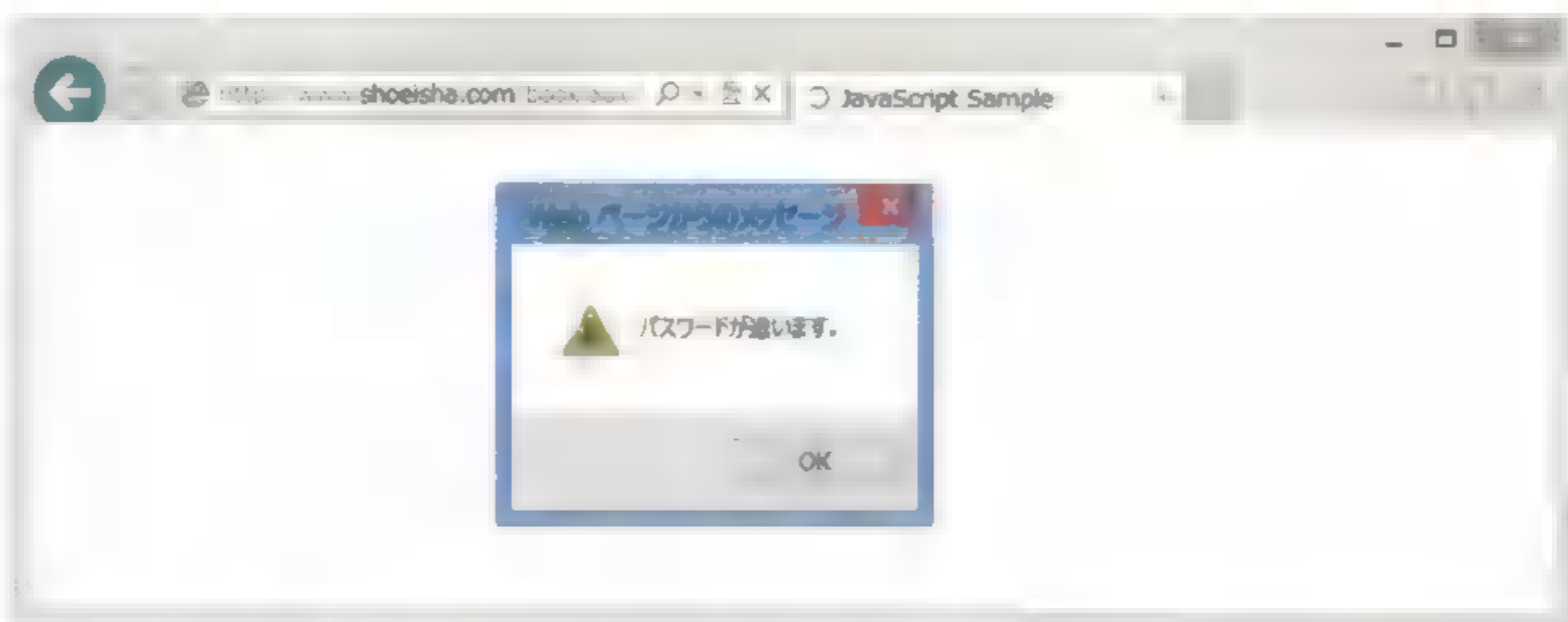
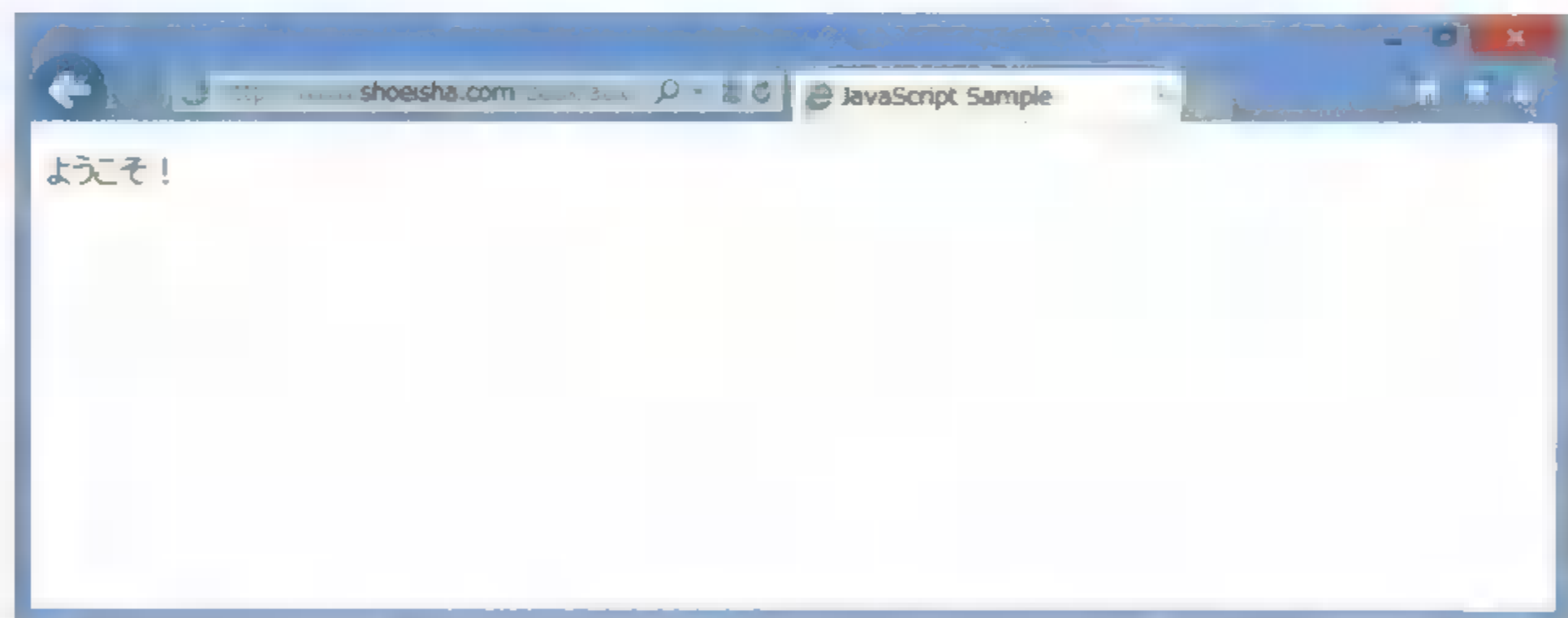
for、while、doなどの繰り返し処理やswitchによる分岐から抜け出します。繰り返し処理の内部でさらに繰り返し処理が行われている場合は、一番内側の繰り返し処理を抜け出します。switchではbreakを記述しないと次のcaseやdefaultの処理に進んでしまいます。1つのcaseの処理を記述したら、必要に応じてbreakで抜け出してください(p.027参照)。

■continue

繰り返しのその回の処理を中断し、処理の先頭に戻って継続します。繰り返し処理の内部でさらに繰り返し処理が行われている場合は、一番内側の繰り返し処理の先頭に戻ります。

サンプルでは3回パスワードチェックを繰り返します。ただし、途中でパスワードが一致した場合はbreakで繰り返し処理を抜け出してページを表示しています。

Source

```
check = false;
for(i = 0; i <3; i ++){ //処理を3回繰り返す
    pass = prompt("パスワードを入力してください。", "********");
    if(pass == "password"){
        check = true;
        break; //パスワードが一致したらfor構文を抜け出す
    }
    alert("パスワードが違います。");
}
if(check == false) //3回の処理後も一致していなければnogood.htmlへ移動
    location.href = "nogood.html";
```

パスワードが異なるとメッセージが表示されます

for構文を繰り返します

3回繰り返す前にパスワードが合致すれば、breakでfor構文を抜け出します

オブジェクトを扱う

構文	説明
▲ = new ★(◆, ◆ … , ◆)	新しいオブジェクトを作成
delete ●	オブジェクトを削除
with(■) ▼	オブジェクト名を省略して処理を記述
this	参照中のオブジェクトを示す

▲……新しいオブジェクトの名前
★……そのオブジェクト作成用の関数
◆……引数【省略可】
●……オブジェクト名
■……オブジェクト名
▼……処理

オブジェクトを作成したり削除したりする制御構文です。

■newステートメント

オブジェクトを作成します。newに続けてオブジェクト作成用の関数を呼び出します。よく使われるものには、Array(配列)、Date(日付)、Image(画像)、Object(オブジェクト)などがあります。作成用の関数を自分で定義すれば、独自のオブジェクトを作成することもできます。

■delete演算子

オブジェクトを削除します。

■withステートメント

指定したオブジェクトに対する処理を行います。withを使用すると、オブジェクト名の記述を省略できます。

■thisステートメント

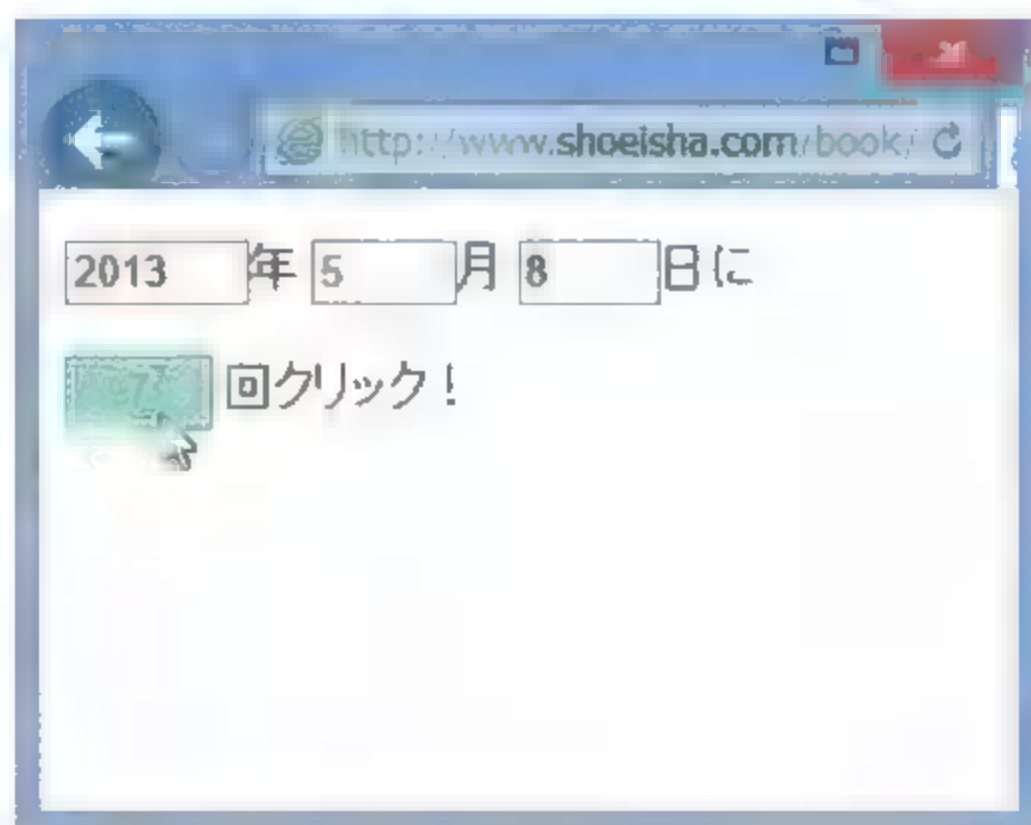
イベントハンドラ内などで参照中のオブジェクトを示すことができます。

まずtodayという日付オブジェクトを作成し、document.form1というフォーム（オブジェクト）のそれぞれの入力欄に、年、月、日を表示しています。さらにボタンをクリックした際にthisステートメントによって関数counterにオブジェクトを渡しています。これによってクリック時にボタンの表面の文字列を書き換える際、document.form1.b1という記述を省略しています。

Source

```
today = new Date(); //日付オブジェクトを作成
myForm = document.getElementById("form1");
with(myForm){ //document.form1オブジェクトについて以下の処理を行う
    text1.value = today.getFullYear();
    text2.value = today.getMonth() +1;
    text3.value = today.getDate();
}
kaisuu =0;
function counter(obj){
    kaisuu++;
    obj.value=" " + kaisuu + " "; //objはdocument.form1.b1の代わり
}
```

Source

```
<body>
<form id="form1">
    <p>
    <input type="text" name="text1" size="4" />年
    <input type="text" name="text2" size="2" />月
    <input type="text" name="text3" size="2" />日に
    </p>
    <input type="button" name="b1" value=" 0 " onclick="counter(this)" />
    回クリック！
</form>
<script type="text/javascript" src="object.js"></script>
</body>
```

今日の日付が表示され、ボタンをクリックした回数がボタン表面に表示されます

BASIC.13

関数

```
function ★(◆){
          ▲
}
```

★……関数名
◆……引数【省略可】
▲……定義する内容

プログラムの中で繰り返し行われる計算や作業をまとめ、何度でも繰り返し使えるようにしたものを関数と言います。JavaScriptでは、あらかじめ用意されている関数のほかに、ユーザーが独自に関数を定義できます。一度関数として定義しておけば、簡単に一連の処理を呼び出して実行させることができます。

関数は次のように定義します

■関数名

functionに続けて任意の関数名を、p.008のような命名規則に沿って指定します。その関数が、どのような処理を行っているのかがわかりやすい名前を付けておくとよいでしょう。本書では任意に付けられる関数名を*msg1* のように斜体で記しています。

■引数

引数とは関数内での処理に必要な数値や文字列などの情報で、この引数の値を変えれば同じ関数でも処理結果を変えることができます。処理に引数が必要な場合は、渡された引数を関数が呼び出された時に受け取るための変数を()内に指定します。「,」(カンマ)で区切れば複数の引数を指定可能です。引数が必要ないときは、空のままにしておきます。

■定義する内容

関数で実際に定義する内容は{ }の間に記述します。関数の中から他の関数を呼び出すこともできます。

次の関数msg2はダイアログを表示した後、関数msg3を呼び出すという2つの動作を行うものとして定義されます。

```
例：function msg2(){
            alert("このボタンはonclickによって呼び出されました。");
            msg3();
    }
```

■return

returnは関数の処理を終了し、呼び出し元に処理を戻します。returnの後に値(戻り値)を指定すると、呼び出し元にその値を返すことができます。

```
例：function myFunc(){
            alert("OK");
            return 1;
    }
    i = myFunc(); // iには1が入る
```

■関数の呼び出し

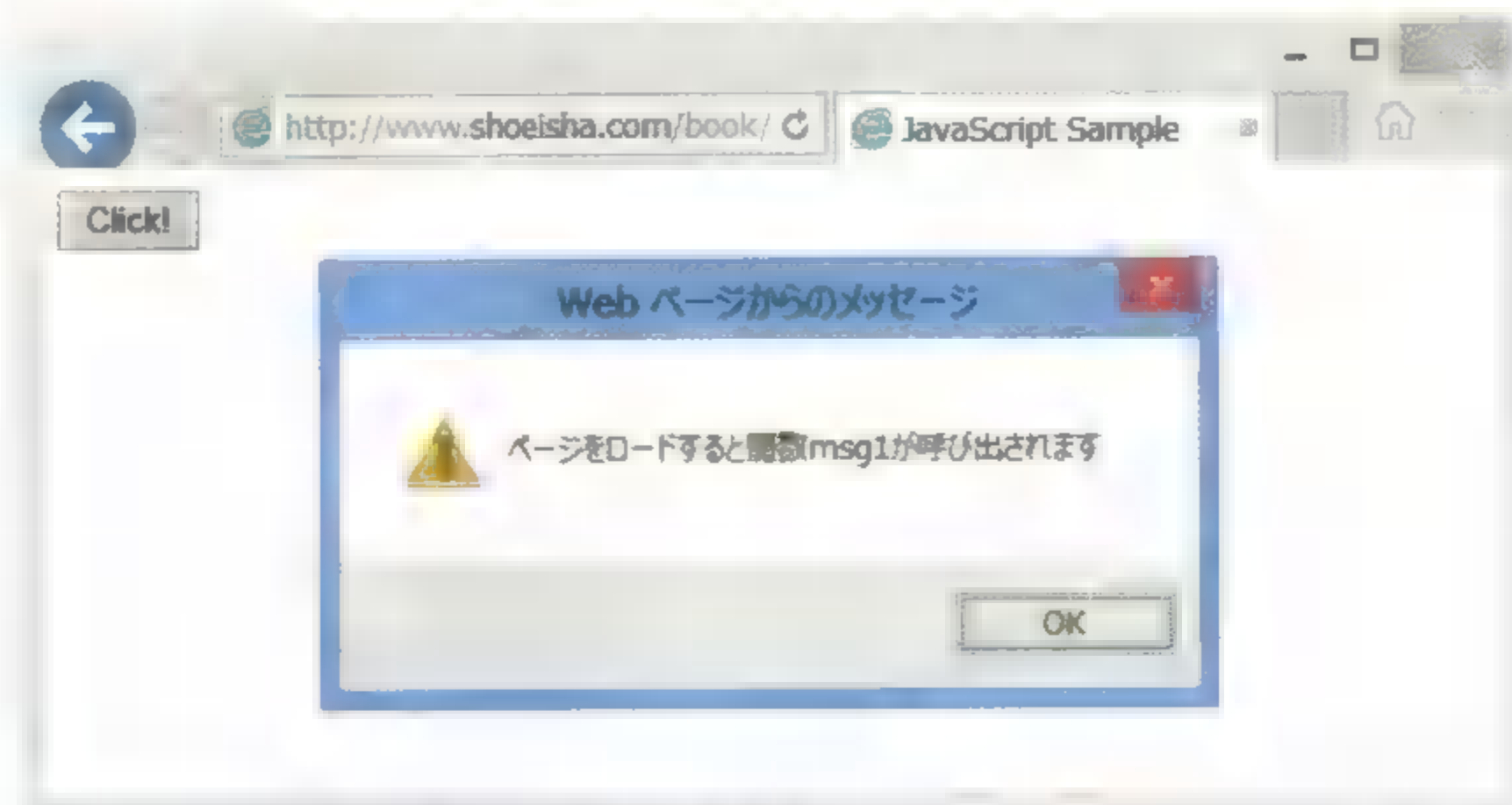
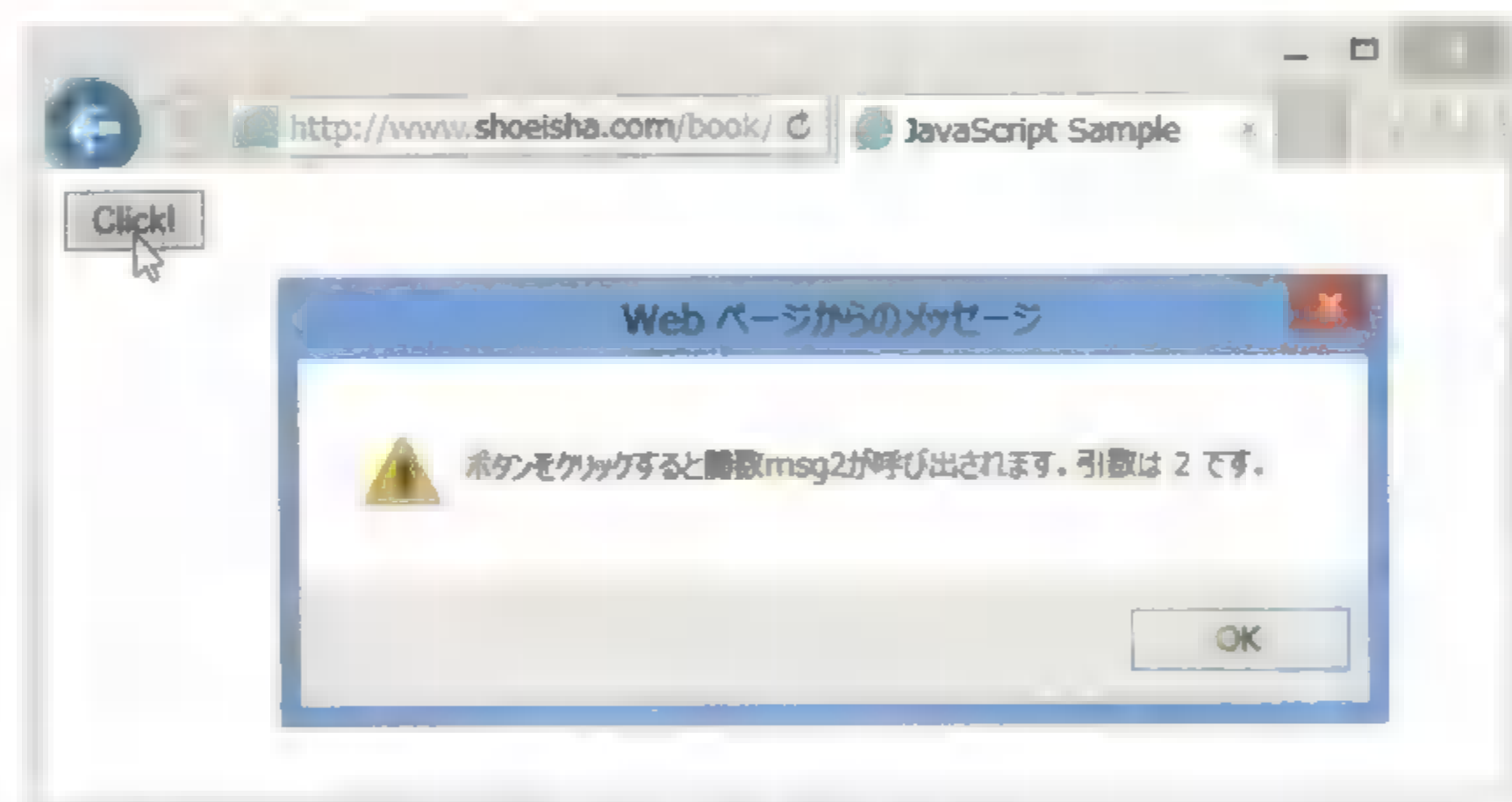
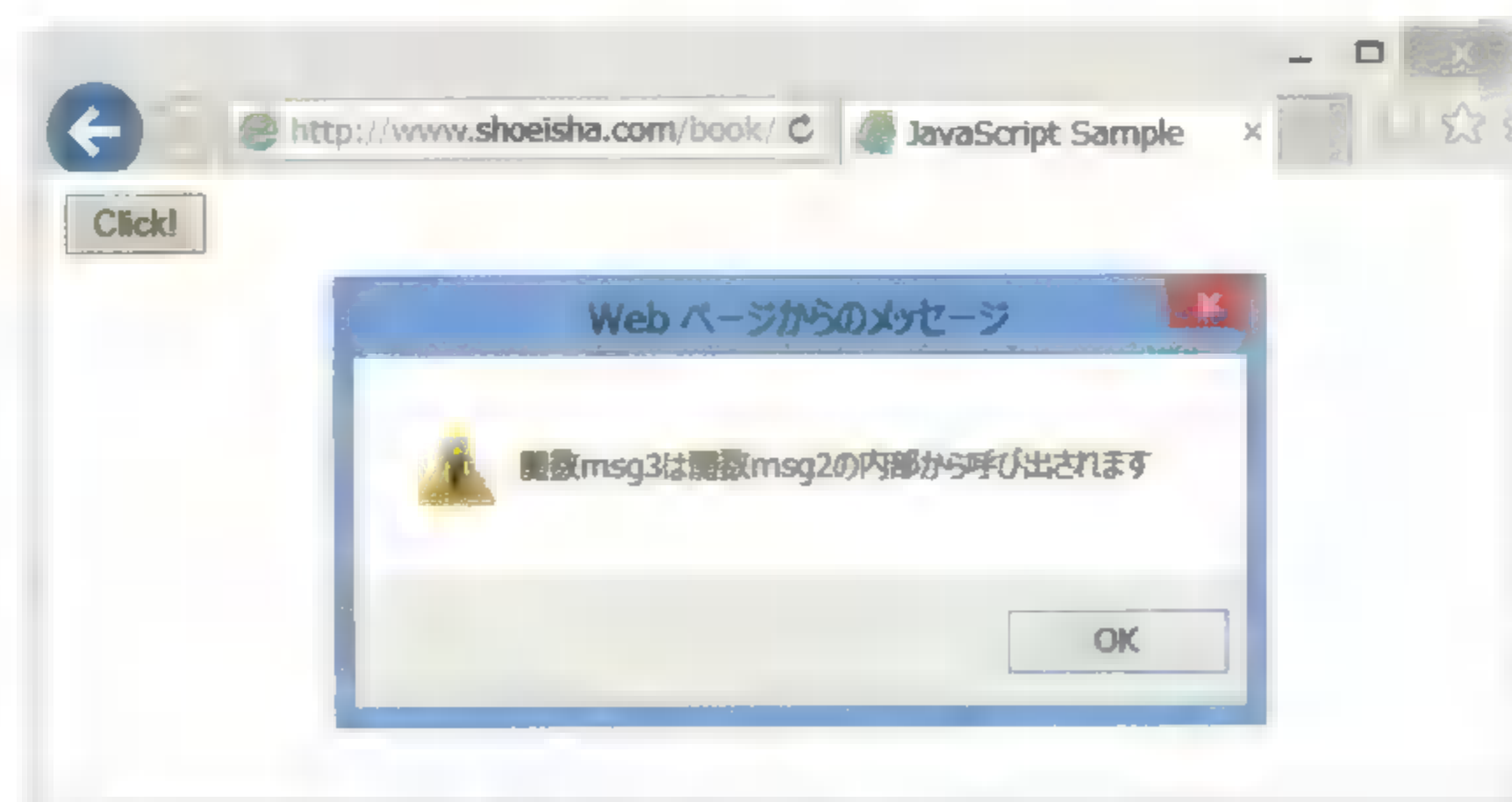
functionは関数を定義するだけなので、スクリプトが読み込まれただけでは実行されません。実際に動作させるにはイベントやほかの関数によって関数が呼び出される必要があります。これを関数の呼び出しと言い、関数名()のように記述します。

Source

```
function msg1(){
    alert("ページをロードすると関数msg1が呼び出されます");
}
function msg2(a){
    alert("ボタンをクリックすると関数msg2が呼び出されます。引数は " + a + " です。");
    msg3();
}
function msg3(){
    alert("関数msg3は関数msg2の内部から呼び出されます");
}
```

Source

```
<body onload="msg1()">
<form id="sample_form">
    <input type="button" value="Click!" onclick="msg2(2)" />
</form>
</body>
```

サンプルではまずページの読み込みが完了したとき(onloadイベント発生時)に関数msg1が実行されます。ボタンがクリックされたときには関数msg2が実行されます。関数msg2の中では関数msg3の呼び出しが行われるので、続けて関数msg3が実行されます。

ページの読み込み時。関数msg1が実行されます

[Click!]ボタンをクリックすると、関数msg2が実行されます

関数msg2から呼び出された関数msg3が続けて実行されます

BASIC.14

DOM

DOMとはDocument Object Modelの略称で、HTML/XHTMLやXMLのためにW3Cによって標準化された規格です。特定のプログラム言語を対象としたものではなく、さまざまなスクリプトやプログラム言語からHTML/XHTML文書やXML文書にアクセスして操作する手段を定義しています。

DOMでは文書を構成する各要素をツリー構造で扱います。このDOMツリーを構成する要素はノードと呼ばれます(要素ノード)。また、DOMではHTML/XHTMLの要素だけでなく属性や要素内容のテキストなどもノードとして扱われ(属性ノード、テキストノード)操作の対象になります。

階層構造をとっている点から、上位の階層のノードは親ノード、下位の階層のノードは子ノード、子孫ノードと表現することもできます。

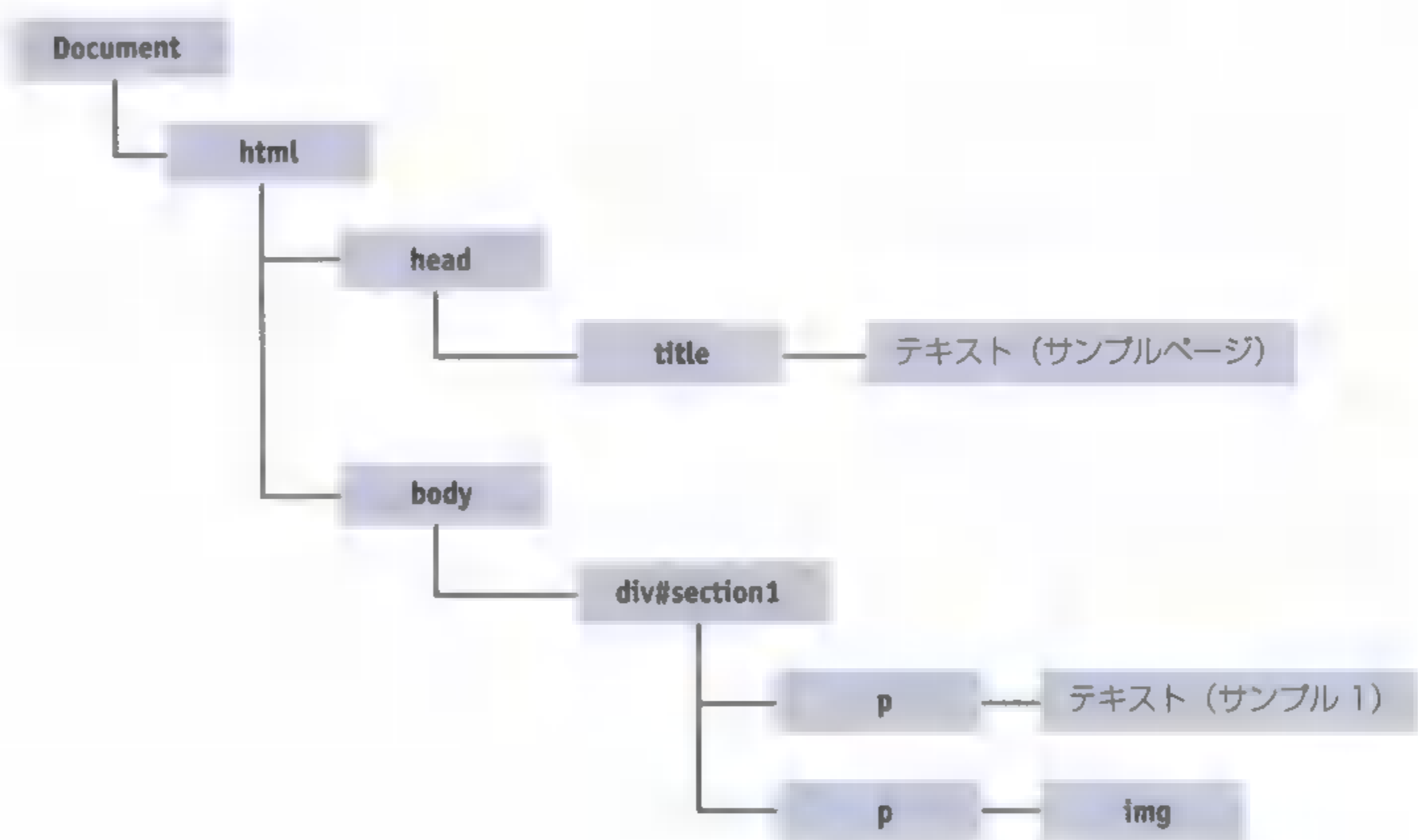
DOMを利用すれば、HTML/XHTMLの要素をすべてJavaScriptのオブジェクトとして操作し、ページの表現をその場でさまざまに変更することが可能になります。

DOMの仕様は現在レベル3まで公開されていますが(2013年5月現在)、その対応状況は各ブラウザやバージョンによって異なっています。本書では、一般的なブラウザが現時点でほぼ対応している基本的なプロパティやメソッドに絞って解説しています。

Source

```
<html>
<head>
<title>サンプルページ</title>
</head>
<body>
<div id="section1">
<p>サンプル1</p>
<p><img src="sample.jpg" alt="解説図" /></p>
</div>
</body>
</html>
```

上記のようなXHTMLファイルをDOMツリーで表すと次のような構造になります。

BASIC.15

Ajax

近年JavaScriptが再度注目を集めている理由の1つに、Googleマップ(ビューの移動や縮尺の変更、衛星写真への切り替えを行う際、地図の再読み込みを必要としない機能を持つ)やGoogleサジェスト(検索窓に文字を入力すると、次の入力を予想し、リアルタイムで候補になりうる言葉を表示する機能)などで利用されているAjaxの人気があります。

Ajaxとは、Asynchronous JavaScript + XML(非同期的なJavaScript + XML)の略語であり、この言葉を提唱したJesse James Garrettによれば、次のように定義されています。(Ajax: A New Approach to Web Applications　http://www.adaptivepath.com/ideas/essays/archives/000385.php)

・XHTMLとCSSを使った標準技術に基づくプレゼンテーション
・Document Object Modelを使ったダイナミックな表示とインタラクティブな仕組み
・XMLとXSLTを使ったデータの変換や操作
・XMLHttpRequestを使った非同期的なデータの取得
・それらをJavaScriptによって結びつける

つまりAjaxという1つの技術があるのではなく、JavaScript、CSS、XML、加えてサーバ側のWebアプリケーション(PHP、Perl、Java、DBなど)のような、既存の技術を組み合わせることで実現されるWebアプリケーションの新しい方法を総じてAjaxと呼んでいるのです。

こうした中で特に注目されたのがXMLHttpRequestというJavaScriptのHTTP通信機能です。XMLHttpRequestはHTTPプロトコルを使ってXMLデータを要求するオブジェクトであり、サーバから非同期的にデータを取得できます。実際にはXMLデータだけでなく、テキストデータも扱えます。

従来のWebブラウザを使ったアプリケーションでは、サーバにデータを送信して処理結果を得るには、ユーザー側でボタンを押すなどして、意図的にWebページを先に進めなければなりませんでした。しかしAjaxでは、サーバと非同期通信を行い、指定したURIからXMLなどのデータを逐一読み込むことで、ページの遷移を行わずに表示内容を動的に変化させることを可能にしています。

BASIC.16

HTML5

HTML5について

HTML5はWebページを記述するのに使用されるHTML(Hypertext Markup Language)仕様の第5版であり、HTML4.01/XHTML1.0の後継として位置づけられています。単に最新の改訂版であるというだけではなく、この間のWeb環境の変化に対応する、大規模な変更が盛り込まれたものとして、大いに期待されています。

■狭義のHTML5

狭い意味でのHTML5は、HTML4.01/XHTML1.0に対する、要素の増減や文法の変化を含みますが、この方面では、header要素やfooter要素、article要素やsection要素など、ヘッダー部や記事や節といった、文書の意味的なまとまりを表す要素が追加され、逆に、font要素などの、視覚的な表示を直接指定する要素が削られました。これは、文書を意味によって構造化し、外観はCSSによって規定するという従来からの方向性にのっとったものです。

一方では、プラグインに依存することなく、HTML標準自体が、動画や音声などを表示することができるように、専用の要素としてvideo要素やaudio要素が追加されました。これには、ブラウザでの高度な表現が求められる中、特定のベンダーの技術や意向に左右される状況は望ましくないとの判断があります。もっとも、対応フォーマットについてはまだ紆余曲折が予想される状況です。

■広義のHTML5

広い意味でのHTML5には、CSS3標準や、スクリプトから操作するさまざまな新APIなどが含まれます。これには、CSSアニメーション、CSSでの3D変形操作、Webフォント、モバイルデバイス対応ＡＰＩ、オフライン操作、ネットワーク通信、SVGやCanvasなどによるグラフィックス操作といったものが挙げられます。HTML5のうち、本書が取り扱うのは主にこの中でもJavaScriptが関係する部分になります。

HTML5登場の背景

では、HTML5はどのような文脈の中で登場してきたのでしょうか。

HTML5以前にはWeb標準を策定する機関であるW3CはHTMLをXMLとして再構成するXHTMLを推進しており、XHTML1系の後継としてXHTML2.0仕様の策定を目指していました。しかし、完全なXMLベースへの移行は、すでに普及した仕様であるHTML4.01との互換性を損なう面があり、また、ブラウザによる高度な表現や、Webアプリケーションの様々な要求にもこたえるものではありませんでした。

そこでW3Cとは別に、Apple、Mozilla、Operaなどが中心になってWHATWG（The Web Hypertext Application Technology Working Group）という[illegible]が独自に立ち上げられ、高機能なWebアプリケーションのニーズに対応することを目指した、新しいWeb標[illegible]（Web applications 1.0）の策定が開始されました。この独自に策定された標準は、WHATWG自身からの強い働きかけもあって、2007年にはついにW3Cの公式な標準化プロセスの対象となり、以後、両者の協調のもとに開発が進められ、現在の「HTML5」へと至っています。

XML/XHTMLを推進するというW3Cの方針の[illegible]あったのは、[illegible]味的にタグ付けされたXML/XHTML文書への移行によって、その「意味」を「[illegible]する」Webを実現しようという壮大な構想でした。これをセマンティック・ウェブと呼びます。しかし市場のニーズはむしろ高機能なインターフェイスや高度な表現力のほうに傾き、なかなか構想が進展しているとはいえない状況でした。

そこにHTML5が登場し、結果としてそれまでW3Cが推進していたXHTML2.0は2009年に放棄されることになりました。とはいえ、XHTMLが完全に廃止されたわけではなく、HTML5では、HTMLでの記法とXHTMLでの記法との両方が許されることとなりました（XHTML5）。また、HTML5で追加されたarticle[illegible]なども、文書を意味的に構造化しようというセマンティック・ウェブの方向性に沿うものといえます。

HTML5で追加された要素

■ページ構造化の要素

HTML5で追加された主な要素としては、header(目次やロゴなどのヘッダー部分)、footer(著作権表示やサイトマップ、関連リンクなどのフッター部分)、article(独立したひとまとまりの記事)、section(章や節といった文書内のセクション)、nav(リンクなどのナビゲーション)、aside(補足などの付随的な情報)、figure(図版)といった記事の意味的なまとまりを表現するものがまず挙げられます。これらの要素は、実際にウェブ上のページで使われているCSSのクラス名を分析することで決定されました(典型的なのはサイドバーのあるブログサイトです)。

こうしたページを意味的に構造化して表現する要素には、通常のIEなどのブラウザ以外の、たとえば視覚障碍者向けブラウザなどで、それぞれの要素の意味に応じた最適な読み上げを行うことが期待できます。同様に、それぞれの要素の意味が明確な場合、検索エンジンがその情報を利用して、より詳細な検索を行うことも考えられます。

■input要素

ユーザーとの高度なやり取りという意味ではinput要素に加えられた変更も重要です。従来はフォーム入力の際は、シンプルな文字列での入力が基本で、入力内容のチェックもスクリプトで独自に行う必要がありました。HTML5では、数値や、日付、メールアドレス、といった入力内容ごとのタイプが追加され、タイプによってはカレンダーやスライダーなどのグラフィカルな入力補助が提供されるようになります。

■メディア関連要素

ブラウザによる高度な表現という意味では、プラグインなしでの音声・動画メディアの再生のための要素の追加も影響の大きな出来事です。標準的なメディアフォーマットが確定しさえすれば、ページの作成者は、基本的にはユーザーの閲覧環境がWeb標準に合致していさえすれば、提供しようとするメディアをユーザーが再生できるかどうかにそれほど悩まずに済むようになります。

ほかにもさまざまな要素がHTML5では追加されていますが、詳細は本書の姉妹編である『HTML5&CSS3辞典 第2版』をご覧ください。

HTML5 APIとJavaScript

さらに、広い意味でのHTML5には、[illegible]のさまざまなWeb関連の技術も含まれます。一つにはデスクトップからクラウドへの移行という大きな流れに沿った、高機能なWebアプリケーションのための技術、さらにはスマートフォンやタブレットの急速な普及に対応するための、モバイルデバイス特有の機能、たとえば位置情報や各種センサーへアクセスするAPIといったものがあります。

デスクトップ並みの機能を有する高度なWebアプリケーションを実現するためのAPIとしては、オフラインにデータを保存するストレージ技術(WebStorage、Indexed Database)、オフライン時にも同じように動作させるためのキャッシュ技術(App Cache)、ローカルファイルへのアクセス(File API)、ユーザーとの対話を[illegible]させることなくバックグラウンドでの処理を可能にする技術(Web Workers)などが用意され、ブラウザは単なるアプリケーションというよりもランタイム[illegible]のようなものになりつつあります。

グラフィカルな表現能力へのニーズという意味ではXMLでのベクター画像を表現するSVG、JavaScriptによる描画を可能にするCanvas、CSS3による高度な外観の指定やアニメーション、変形操作といったものがあり、スクリプトからの操作もそれぞれ可能になっています。

Ajaxで脚光を浴びた[illegible]関連では、WebSocket、Server-Sent Eventsといった双方向リアルタイム性を重視した通信APIがあり、従来のAjaxと組み合わせることにより、画面のリロードを必要としない動的な画面の書き換えなども可能になっています。

今後ますますブラウザの側の対応が進むことにより、HTML5を利用することで、高度な表現力と、[illegible]準準拠による高い互換性、意味的なタグ付けによる可能性、といったものを両立させたサイトやWebアプリケーションが可能になるものと考えられます。

第2部
JavaScriptリファレンス
JavaScript REFERENCE

DIALOG.01

警告ダイアログを表示したい

★.alert(◆)

★……Windowオブジェクト(ウィンドウ名またはフレーム名)【省略可】
◆……ダイアログに表示する文字列や数値

形式 メソッド

[OK]ボタンのある警告ダイアログを表示するメソッドです。[OK]ボタンがクリックされるか[×]ボタンでダイアログが閉じられるまで、スクリプトは次の処理へ進みません。また、ダイアログが表示されている間はブラウザの操作は行えません。

ウィンドウ名を★に指定した場合、指定したウィンドウにダイアログが表示されます。

文例

```
alert("未入力の項目があります");
```

「未入力の項目があります」という警告ダイアログを表示します。

```
<a href="http://books.ank.co.jp/" onClick="alert('書籍サイトに移動します。')">
```

リンクがクリックされたとき、「書籍サイトに移動します。」という警告ダイアログを表示します。

Column [ダイアログの改行]

ダイアログに表示する文字列を改行したい場合には、以下のサンプルのように「¥n」を挿入します。

```
"1行目の文字列を記入します¥n" + "2行目と¥n" + "3行目"
```

ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照
確認ダイアログを表示したい ･･････････････ P.051
文字入力ダイアログを使いたい ･･･････････ P.052
【SAMPLE】ダイアログを表示する ･･････ P.053

DIALOG.02

確認ダイアログを表示したい

◆ = ★.confirm(▲)

◆……変数([OK]でtrue、[キャンセル]でfalseが代入される)
★……[illegible]Windowオブジェクト(ウィンドウ名またはフレーム名)【省略可】
▲……ダイアログに表示する文字列や[illegible]値

形式 メソッド

[OK]ボタンと[キャンセル]ボタンのある確認ダイアログを表示するメソッドです。いずれかのボタンがクリックされるまでスクリプトは次の処理へ進みません。また、ダイアログ表示時はブラウザの操作は行えません。[OK]ボタンがクリックされた場合はtrue、[キャンセル]ボタンまたは[×]ボタンがクリックされた場合はfalseを値として返します。なお、★を指定した場合、指定したウィンドウにダイアログを表示します。

文例

```
confirm("本当にこれでいいんですね？");
```

確認ダイアログに「本当にこれでいいんですね？」と表示します。

```
if(confirm("株式会社アンクのページに移動しますか？")){
    location.href = "http://www.ank.co.jp/";
}
```

確認ダイアログを表示し、[OK]ボタンがクリックされるとhttp://www.ank.co.jp/ に移動します。

▶ ブラ[illegible]対応表	IE10	IE9	[illegible]	Fx	Chrome	Safari	[illegible]	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照

警告ダイアログを表示したい……………P.050
文字入力ダイアログを使いたい…………P.052
【SAMPLE】ダイアログを表示する………P.053

DIALOG.03

文字入力ダイアログを使いたい

● = ★.prompt(◆, ▲)

●……入力された文字列
★……Windowオブジェクト(ウィンドウ名またはフレーム名)【省略可】
◆……ダイアログに表示する文字列
▲……初期入力されている文字列【省略可】

形式 メソッド

文字入力欄と[OK]ボタン、[キャンセル]ボタンのあるダイアログを表示するメソッドです。いずれかのボタンがクリックされるまで、スクリプトは次の処理へ進みません。また、ダイアログ表示時はブラウザの操作は行えません。

[OK]ボタンがクリックされた場合は入力欄に入力されている文字列、[キャンセル]ボタンまたは[×]ボタンがクリックされた場合はnullを値として返します。初期状態で文字入力欄に何も表示したくない場合は、prompt("test","")のように2つめの引数▲に「""」(ダブルクォーテーション)のみを記述してください。

なお、★を指定した場合、指定したウィンドウにダイアログを表示します。

文例

```
prompt("あなたの名前は？ ", "");
```

名前の入力を求める文字入力ダイアログを表示します。初期状態では入力欄には何も表示しません。

```
pass = prompt("合言葉を入力してください", "山");
```

合言葉の入力を求める文字入力ダイアログ(初期状態では入力欄に「山」と表示)を表示し、入力された文字を変数pass に代入します。

ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

警告ダイアログを表示したい・・・・・・・・・・・・・ P.050
確認ダイアログを表示したい・・・・・・・・・・・・・ P.051
【SAMPLE】ダイアログを表示する・・・・・・・・・ P.053

DIALOG.SAMPLE-01

ダイアログを表示する

警告／確認／文字入力ダイアログを使ったログオンページのサンプルです。ユーザー名が入力され、[ログオン]ボタンがクリックされると、ユーザー名の確認ダイアログを表示します。ユーザー名が未入力の場合は警告ダイアログを表示します。

確認ダイアログにユーザー名が入力され、[OK]ボタンがクリックされると、パスワード入力(文字入力)ダイアログを表示します。なお、[キャンセル]ボタンがクリックされた場合は処理を終えます。

パスワード入力ダイアログにpasswordと入力されると、ユーザー名入りのメッセージを表示します。未入力、もしくはpassword以外の文字列が入力された場合は「パスワードが正しくありません」というメッセージを表示します。また、パスワード入力ダイアログの[キャンセル]ボタンがクリックされると、「パスワード入力を終了します」と表示し、処理を抜けます。

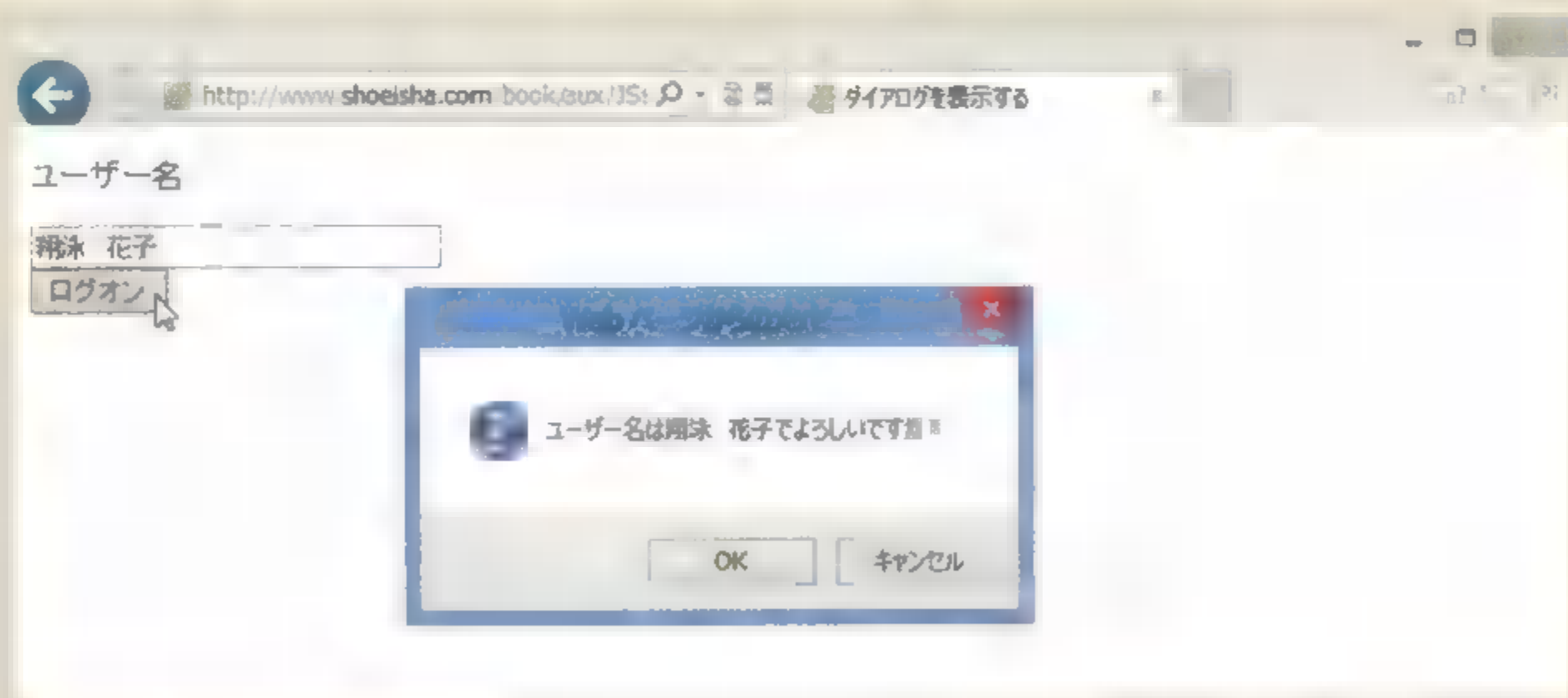
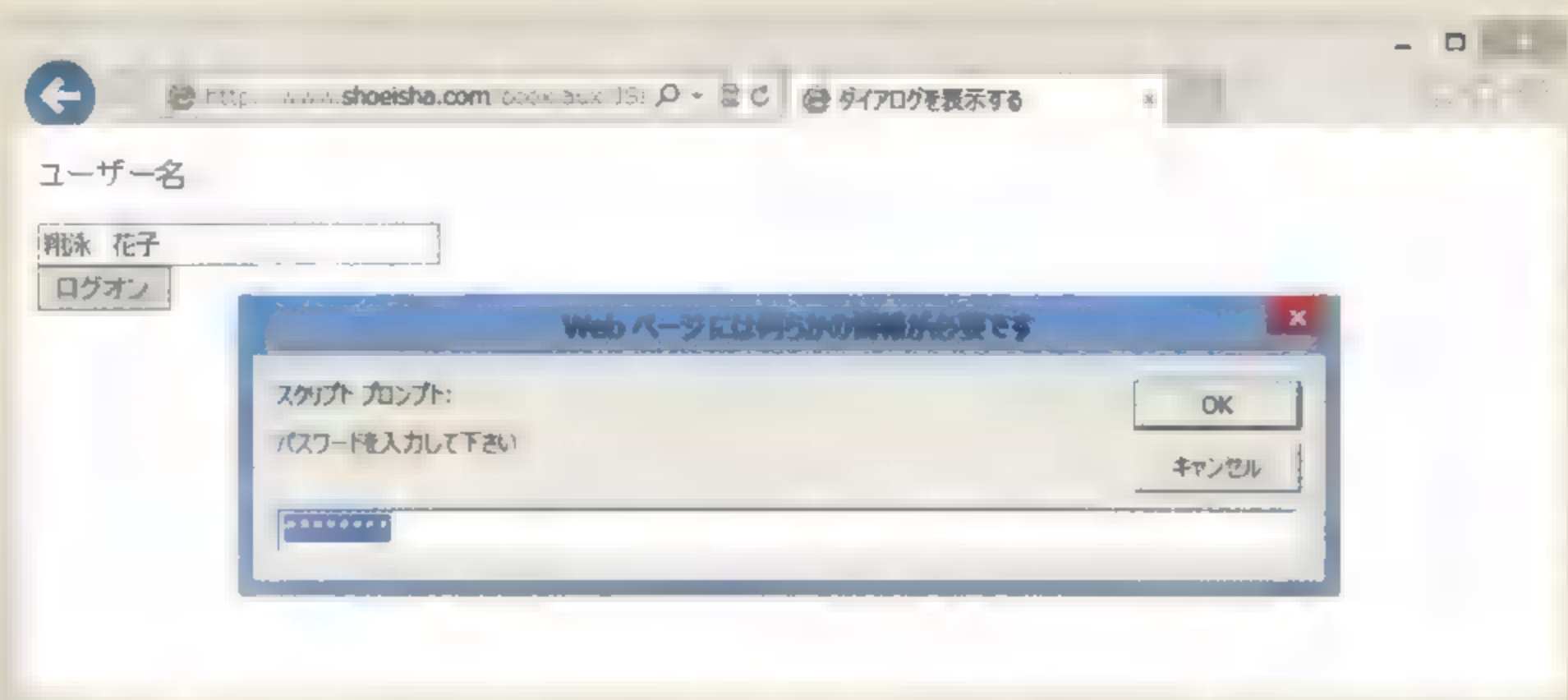
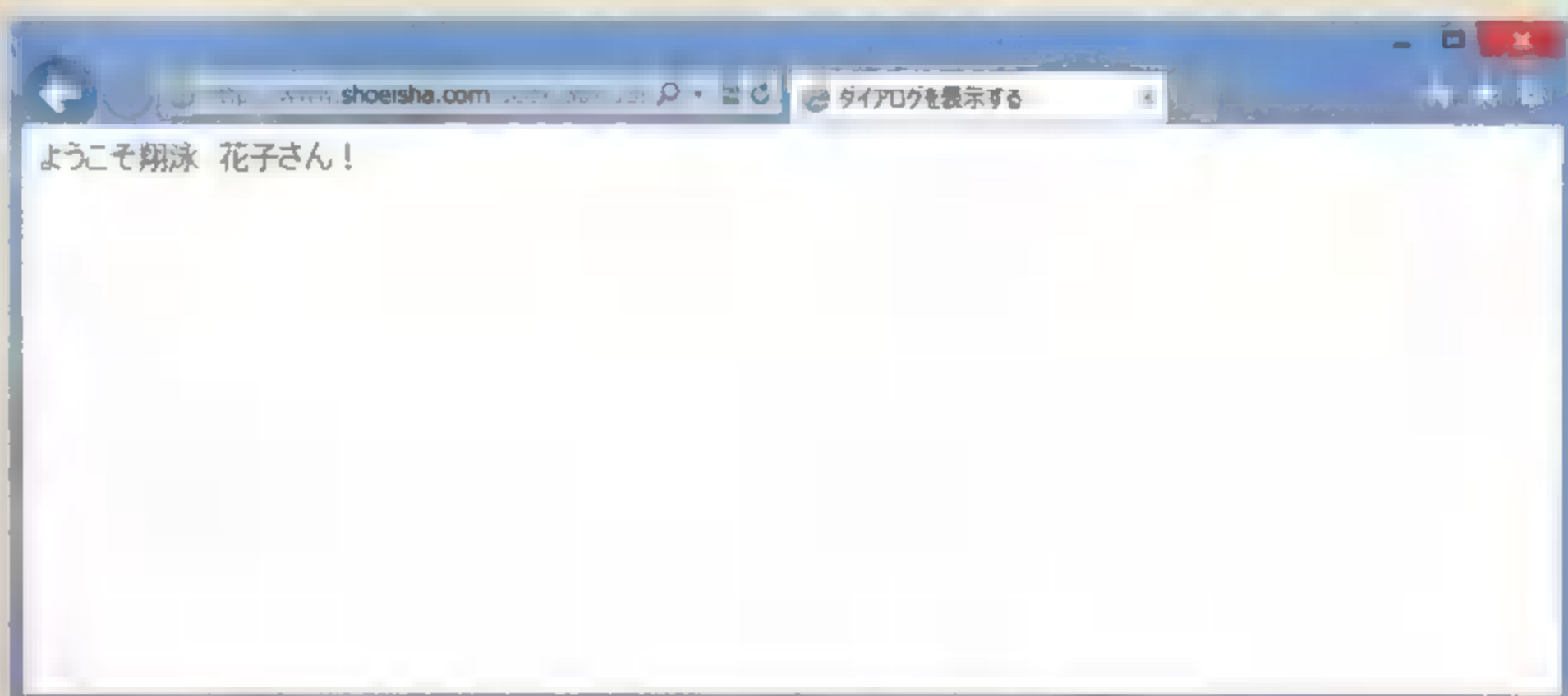
JavaScript

```
//ダイアログを表示させる[illegible]
function show_dialog(){
    var username = document.getElementById("text1").value;

    if(username == ""){ //ユーザー名が未入力の[illegible]
        alert("ユーザー名を入力して下さい。");
        return;
    }

    var res = confirm("ユーザー名は" + username + "でよろしいですか？");
                                            //confirmダイアログの結果をresに代入

    if(res == true){ //「OK」を選択
        var pass = prompt("パスワードを入力して下さい", "********");

        if(pass == null){ ///パスワードが未入力の場合
            alert("パスワード入力を終了します");

        }else if(pass != "password"){ ///パスワードが"password"以外の場合
            alert("パスワードが正しくありません");
        }else{ //正常処理
            document.open();
            document.write("ようこそ", username, "さん！");
            document.close();
        }
```

```
  }
}
```

HTML

```
<body>
  <form action="/cgi-bin/sample.cgi">
  <p>ユーザー名</p>
  <p>
    <input type="text" id="text1" size="30" /><br />
    <input type="button" value="ログオン" onclick="show_dialog()" />
  </p>
  </form>
</body>
```

❶ユーザー名が未入力の場合は警告ダイアログが表示されます

❷ユーザー名を入力し、[ログオン]ボタンをクリックすると、ユーザー名の確認ダイアログが表示されます

❸ユーザー名の確認ダイアログの[OK]ボタンをクリックすると、文字入力ダイアログが表示されます

❹文字入力ダイアログに「password」と入力すると、メッセージが表示されます

参照

alert メソッド ････････････････････ P.050
confirm メソッド ･･････････････････ P.051
prompt メソッド ･･･････････････････ P.052

DOCUMENT.01

ドキュメントを扱いたい

★.document

★……Windowオブジェクト【省略可】

形式 オブジェクト

ページに表示される文字列や画像、フォーム、リンクなどのHTML/XHTMLのドキュメントを制御するオブジェクトです。

文例

```
document.write("JavaScript");
```

ドキュメントに「JavaScript」と書き出します。

```
fgCol = document.fgColor;
```

文字色を変数fgColに代入します。

▶ ブラウザ対応表	IE10	IE9	[illegible]	Fx	Chrome	Safari	Opera	[illegible]	Android
	○	○	○	○	○	○	○	○	○

DOCUMENT.02

ドキュメントをオープン／クローズしたい

ドキュメントの出力を開始
ドキュメントの出力を終了

★……Documentオブジェクト（ドキュメント名）

形式 メソッド

openメソッドとcloseメソッドはドキュメントをオープン／クローズするときに利用します。現在のドキュメントに対して使用すると、open()からclose()の間で指定した内容でドキュメントの内容を置き換えます。

openメソッド

ドキュメントを開いて、書き出し可能な状態にします。

closeメソッド

ドキュメントの書き出しを終了するメソッドです。openメソッドで書き出しを開始したドキュメントは、必ずcloseメソッドで書き出しを終了させてください。終了していない場合はページが読み込み中の状態のままになってしまいます。

文例

newWin.document.open();
ウィンドウnewWinのドキュメントの書き出しを開始します。

newWin.document.close();
ウィンドウnewWinのドキュメントの書き出しを終了します。

▶ブラウザ対応表	IE10	IE9	[illegible]	[illegible]	Chrome	Safari	Opera	[illegible]	Android
	○	○	○	○	○	○	○	○	○

参照
文字列や画像を表示したい …………… P.058
【SAMPLE】文字や画像を表示する ……… P.066

DOCUMENT.03

文字列や画像を表示したい

★.write(◆)　文字列を書き出す
★.writeln(◆)　文字列を書き出して改行する

★……Documentオブジェクト(ドキュメント名)
◆……書き出す文字列(HTMLコード)

形式 メソッド

ドキュメントに文字列を書き出すメソッドです。文字列中のタグはページ上の表示に反映されます。

HTMLソースを直接書き換えるのと同じ効果があるため、ある意味で何でもできてしまうメソッドです。操作を間違えるとHTMLの構造自体が壊れてしまうため、他で代用できる場合は推奨されません。

デバッグ目的で画面表示を行いたい場合は「console.log(文字列)」というメソッドを使うと、ブラウザのデバッグ画面(IE10の場合、F12で開きます)に書き出しを行うことができます。

writeメソッド

引数として与えられた文字列をドキュメントに書き出します。

writelnメソッド

引数として与えられた文字列を書き出した後に改行文字が付加されます。なお、改行を有効にするためには書き出す文字列を<pre>～</pre>タグで囲んでおく必要があります。

文例

```
document.write("こんにちは、" + name + "さん！");
```

変数nameの値の前後に「こんにちは、」と「さん！」をつけ、「こんにちは、XXさん！」と書き出します。

```
document.writeln("<img src='cat1.gif' alt='' />");
```

画像cat1.gifを表示して改行します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

参照
ドキュメントをオープン／クローズしたい・・・P.057
【SAMPLE】文字や画像を表示する・・・・・・・・・P.066

DOCUMENT.04

最終更新日を自動的に挿入したい

★.lastModified

★……Documentオブジェクト（ドキュメント名）

形式 プロパティ

ファイルの最終更新日を表す文字列を返すプロパティで、ページの最終更新日を自動的に表示する場合などに使用します。なお、ブラウザの種類やOS等の設定によって、返される日付文字列の形式は異なります。

文例

```
document.write("Last Update: " + document.lastModified);
```

最終更新日を表示します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

参照

日付を取得したい …………………… P.182
時刻を取得したい …………………… P.185
【SAMPLE】ドメイン情報を取得する ……… P.067

DOCUMENT.05

ドメイン名を参照したい

★.domain

★……Documentオブジェクト(ドキュメント名)

形式 プロパティ

domainプロパティはドキュメントが置かれているドメイン名(www.ank.co.jpなど)を返します。複数の場所に同じページを置くような場合、ドメインによってリンクやバナー広告を変更するなどドメインの判断が必要なときに使用します。

文例

```
if(document.domain == "www.ank.co.jp"){
    alert("アンクのWebサイトです");
}
```

ドメイン名がwww.ank.co.jpの場合、「アンクのWebサイトです」というダイアログを表示します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照

ページのロケーション情報を参照したい …… P.231
URIを参照/設定したい …… P.226
【SAMPLE】ドメイン情報を取得する …… P.067

DOCUMENT.06

ドキュメントのタイトルを参照したい

★.title

★……Documentオブジェクト（ドキュメント名）

形式 プロパティ

ドキュメントのタイトル（HTML/XHTML文書中<title>～</title>タグで囲まれた部分）を参照／設定します。

文例

```
document.write("タイトル: " + document.title);
```
ドキュメントのタイトルを表示します。

```
docTitle = document.title;
```
ドキュメントのタイトルを変数docTitleに代入します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照 【SAMPLE】ドメイン情報を取得する ・・・・・・・ P.067

DOCUMENT.07

選択されている文字列を調べたい

★.getSelection()
◆.getRangeAt(参照番号);

★……DocumentオブジェクトまたはWindowオブジェクト
◆……Selectionオブジェクト

形式 メソッド

Documentオブジェクトの getSelectionメソッドは、マウスなどで選択されている文字列を返します。

HTML5では、WindowオブジェクトのgetSelectionメソッドは、文字列ではなくSelectionオブジェクトを返します。一部のブラウザでは、DocumentオブジェクトのgetSelectionメソッドもSelectionオブジェクトを返すように変更されています。ブラウザでの差異に対応するには、window.getSelection()を使用します。

Selectionオブジェクトは現在の選択範囲全体を表し、個別の選択範囲をRangeオブジェクトが表します。選択範囲が複数に分断されているとき(コントロールキーを使って選択した場合など)は、一つのSelectionに対して、複数のRangeが属する形になります。SelectionからRangeを取り出すにはgetRangeAtメソッドを使用します。

getRangeAt(▲)
選択範囲内の、指定された参照番号▲のRangeオブジェクトを返します。

また、以下のメソッドを使用すると、1つの連続した単位の選択範囲(Range)を、全体の選択範囲(Selection)に追加したり削除したりすることができます。

addRange(■)
指定のRangeオブジェクト■を全体の選択範囲に追加します。

removeRange(■)
指定のRangeオブジェクト■を全体の選択範囲から削除します。

selectAllChildren(●)

指定された要素●の子要素すべてを全体の選択範囲に追加します。

deleteFromDocument()

現在の選択範囲全体をドキュメントから削除します。

Selectionオブジェクトや Rangeオブジェクトが表す選択範囲の文字列を取得するには、オブジェクトのtoStringメソッドを使用します。また、Selectionオブジェクトや Rangeオブジェクトには文字単位での選択範囲の開始位置と終了位置を求めるプロパティも存在します。これについてはサンプルを参照ください。

文例

```
var st = document.getSelection();
```

選択された文字列を変数stに代入します。

```
var selection = window.getSelection();
```

Selectionオブジェクトを取得します。

```
var range = selection.getRangeAt(0);
```

1番目の選択範囲を表すRangeオブジェクトを取得します。

```
selection.removeRange(range);
```

1番目の選択範囲の内容をドキュメントから削除します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	×	○	○	○	○	○	○

参照

【SAMPLE】[illegible]されている文字列を調べる ‥P.068

DOCUMENT.08

クッキーを使いたい

★.cookie

★……Documentオブジェクト（ドキュメント名）

形式 プロパティ

cookieプロパティはクッキーの文字列を参照／設定します。プロパティの値はクッキーの文字列になります。

クッキーの文字列は「キーワード1=値1; キーワード2=値2; ……」という形式になっています。そのため、書き込む際には「=」（イコール）や「;」（セミコロン）を付加し、逆に読み込む際にはそれらを切り離して値を取得する作業が必要です。また、「キーワード1=値1;」「キーワード2=値2;」……のように文字列を順に設定すると、上書きではなく追加ができます。

クッキーには有効期限を設定できます。これにより、「1日間だけ有効なクッキー」といった設定も可能です。有効期限を設定しない場合は、ドキュメントが閉じられるまで有効です。有効期限に過去の日時を指定すると、クッキーを削除できます。

ただし、保存できるクッキーの数やサイズには限界があり、それを超えた場合は古いものから削除されていきます。なお、ユーザーのブラウザがクッキーを無効にするよう設定されている場合は、クッキーへの書き込みは行えません。

文例

```
document.cookie = "name=ank; Tue, 31-Dec-2014 23:59:59 GMT";
```

有効期限付きでクッキーに書き込みます。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照

文字列をエンコード／デコードしたい ……P.250
日付や時刻を扱いたい……P.180
【SAMPLE】クッキーを使う……P.070

DOCUMENT.09

アプレットやプラグインを参照したい

★.applets	Javaアプレット数を参照
★.embeds	プラグイン数を参照
★.plugins	プラグイン数を参照

★……Documentオブジェクト（ドキュメント名）

形式 プロパティ

ドキュメント内のJavaアプレットやプラグインのオブジェクトの配列を参照します。

appletsプロパティ

ドキュメント内の、<applet>タグで定義されたJavaアプレットのリスト（配列）を参照します。

embedsプロパティ

ドキュメント内の、<embed>タグで定義されたプラグインのリスト（配列）を参照します。

pluginsプロパティ

ドキュメント内のプラグインのリスト（配列）を参照します。

文例

```
alert("Javaアプレットの数は" + document.applets.length + "です。");
```

ドキュメントに含まれるアプレットの総数をダイアログに表示します。

Column lengthプロパティについて

lengthは配列の大きさを取得するプロパティです。以下のようにすると、ドキュメント上の画像オブジェクトのname属性を列挙できます（p.30）。

```
for(i=0; i<document.images.length; i++){
  document.write(document.images[i].name,"<br />");
}
```

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

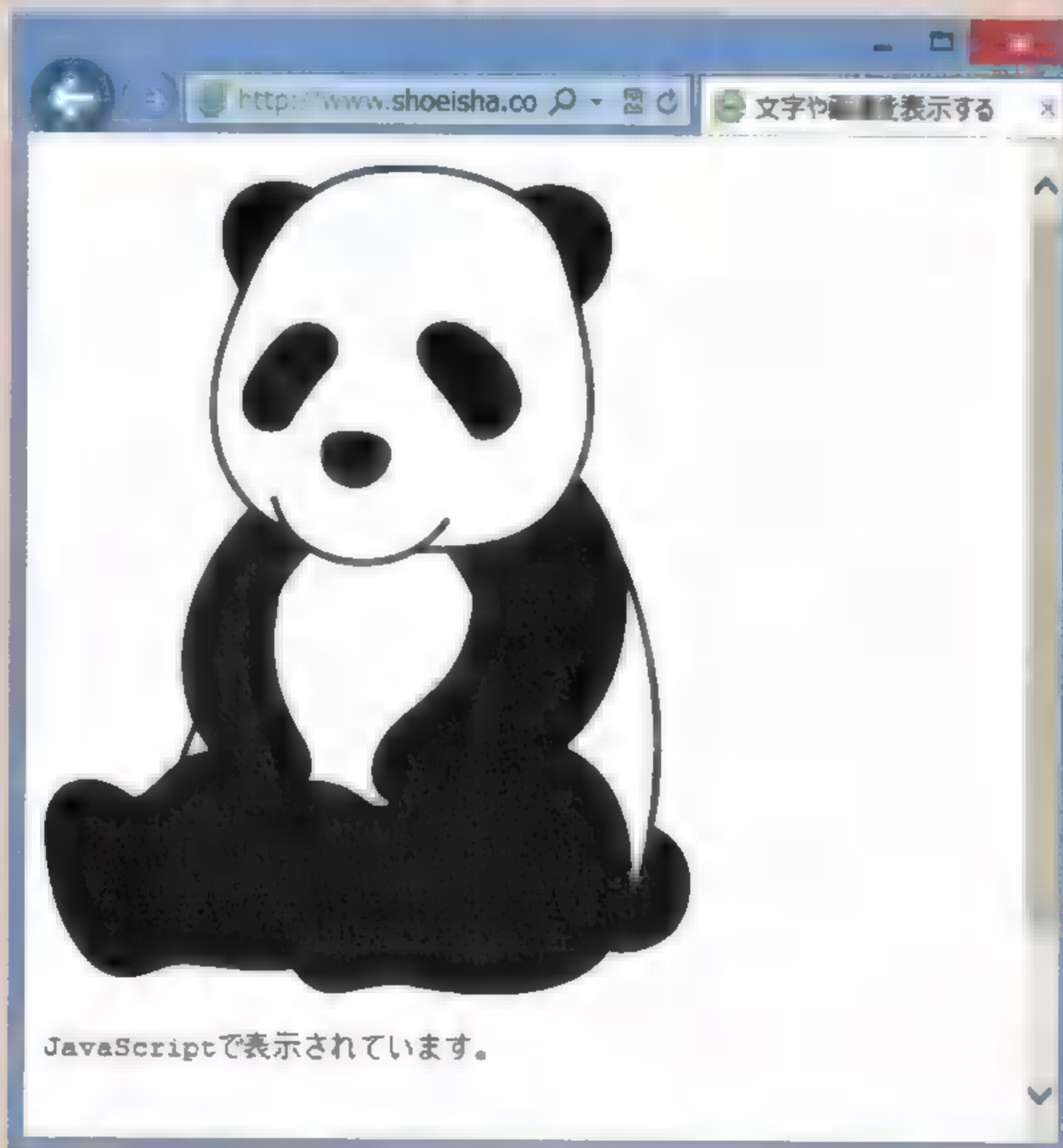
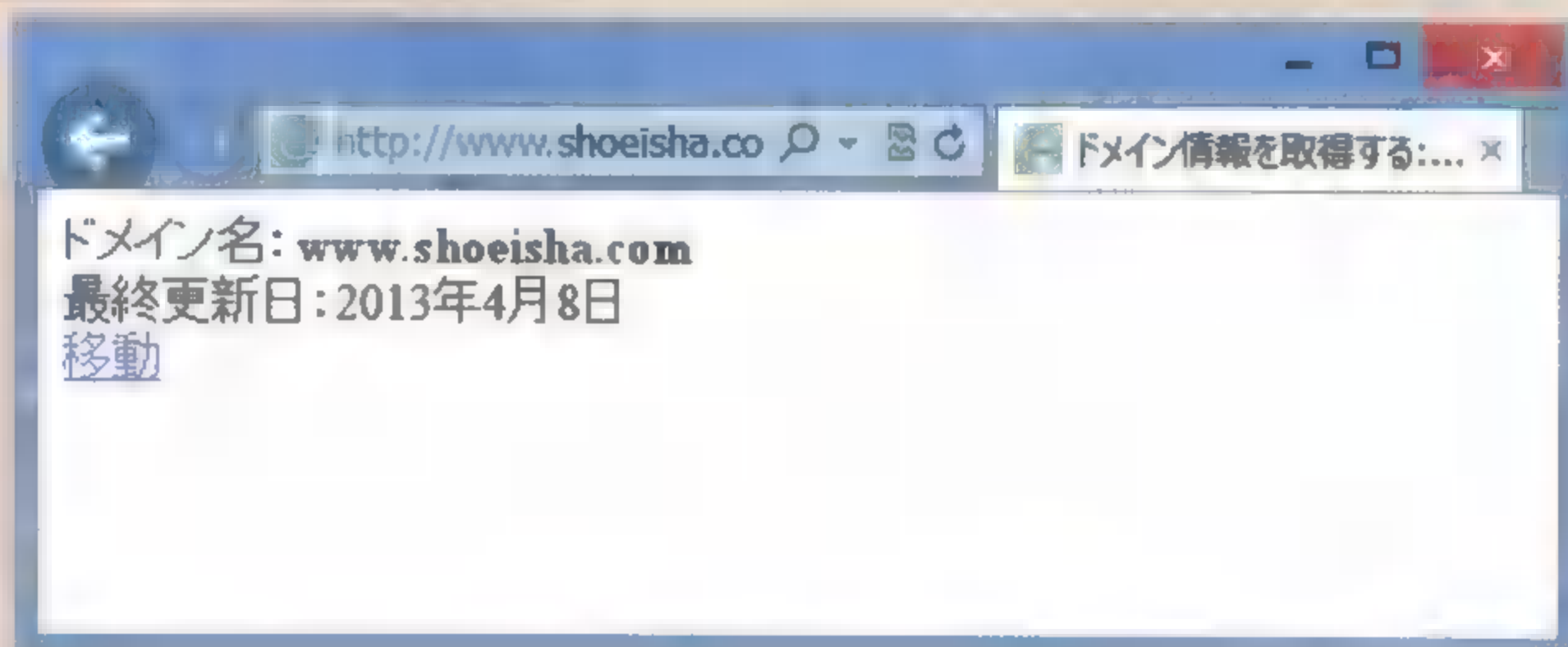
ドキュメント

DOCUMENT.SAMPLE-01

文字や画像を表示する

SAMPLE SCRIPT

JavaScriptで文字や画像を表示します。

JavaScript

```
document.open();
document.writeln("<pre>");
document.writeln("<img src='images/panda.png' alt='' />");
document.writeln("<pre/>");
document.write("JavaScriptで表示されています。");
document.close();
```

Internet Explorer

JavaScriptで表示されています。

open メソッド ······ P.057
close メソッド ······ P.057
write メソッド ······ P.058
writeln メソッド ······ P.058

DOCUMENT.SAMPLE-02

ドメイン情報を取得する

SAMPLE SCRIPT

ドメイン、最終更新日の情報を参照するサンプルです。ドメインを取得し、タイトルバーに表示されるタイトルの最後に付け足しています。最終更新日は、lastModifiedプロパティで取得した日付オブジェクトからgetFullYear、getMonth、getDateメソッドでそれぞれ年月日を取得し、●●●●年●●月●●日の形式で表示しています。

JavaScript

```
var update = new Date(document.lastModified); //最終更新日を取得
var year = update.getFullYear();
var month = update.getMonth();
var date = update.getDate();
document.title += ":"+ document.domain; //ページのタイトルにドメイン名を付け足す
document.write("ドメイン名："+ document.domain +"<br />");
document.write("最終更新日："+year+"年"+month+"月"+date+"日<br />");
document.write("<a href='http://●●●/myweb/document_domain/' >移動</a>");
```

lastModified プロパティ ・・・・・・・・・・・・・・・ P.059
domain プロパティ ・・・・・・・・・・・・・・・・・ P.060
title プロパティ ・・・・・・・・・・・・・・・・・・・ P.061

DOCUMENT.SAMPLE-03

選択されている文字列を調べる

マウスで選択された文字列をフォームの入力欄に書き出すサンプルです。onmouseupイベントを利用し、マウスボタンが離されるたびに選択文字列を取得します。

HTML5版APIが利用できる場合は、Selectionオブジェクトを利用して、どの要素の何文字目からどの要素の何文字目まで選択範囲が存在するかを表示します。Rangeオブジェクトを使う場合は、startContainerが開始要素、startOffsetが開始位置、endContainarが終了要素、endOffsetが終了位置になるほかは、サンプルと同様です。

JavaScript

```
//選択した文字列をテキストボックスに表示させる関数
function selectStr() {
    var textElem = document.getElementById("text1");
    if (window.getSelection) {
        // HTML5 API
        var sel = window.getSelection(); // Selectionオブジェクト
        var str = "「" + sel.anchorNode.nodeValue + "」の"; // 選択の起点要素
        str += sel.anchorOffset + "文字目から"; // 要素内の開始位置
        str += "「" + sel.focusNode.nodeValue + "」の"; // 選択の終点要素
        str += sel.focusOffset + "文字目まで"; // 要素内の終了位置
        document.getElementById("info").innerHTML = str;
        textElem.value = sel.toString(); // 選択文字列はtoString()で取り出す
    } else if (document.getSelection) {
        // 旧API
        textElem.value = document.getSelection();
    } else {
        // 非対応ブラウザ
        textElem.value = "参照不可";
    }
}
```

HTML

```
<body onmouseup="selectStr();">
<p>
    <span id="part1">選択されている</span><span id="part2">文字列を</span><span
id="part3">取得したい</span>
</p>
```

```
<form>
  <p>選択文字列：<input type="text" id="text1" /></p>
  <p id="info"></p>
</form>
</body>
```

選択した文字がフォームの入力欄に表示され、開始要素、開始位置、終了要素、終了位置が表示されます。

Internet Explorer 8

選択されている文字列を調べる - Windows Internet Explorer

選択されている文字列を取得したい

選択文字列: 参照不可

getSelectionメソッド非対応のブラウザでは「参照不可」と表示されます。

getSelection メソッド …………………… P.062

DOCUMENT.SAMPLE-04

クッキーを使う

初回訪問時に名前が入力されていると、次回以降は「ようこそ(名前)さん！」と表示するサンプルです。まず、ページが読み込まれると、loadPage関数が呼び出されます。getCookie関数でNAMEというキーで値があるかないかを調べます。なかった場合(初回訪問時)は名前の入力を促すテキスト入力フィールドを表示します。[OK]ボタンがクリックされると、入力された名前をsetCookie関数でNAMEというキーでクッキーにセットします。2回目以降の訪問ではクッキーから取得した値を使って「ようこそ(名前)さん！」と表示します。表示の振り分けはinnerHTMLを使用し、HTML文書内の<div id="div">タグ内のHTMLを書き換えることで実現します。

以下、setCookie関数、getCookie関数について解説します。

●setCookie関数

クッキーに値を書き込む関数です。

```
① function setCookie(keyname, val){
②       var tmp = keyname + "=" + escape(val) + ";";
③       tmp += "expires = Mon,31-Dec-2015 23:59:59;";
④       document.cookie = tmp;
⑤ }
```

①引数としてkeyname、valを受け取ります。keynameはクッキーに保存するキー名、valは保存する値です。

②変数tmpに「keyname = (文字コードに変換した)val;」を代入します。

※クッキーには日本語をそのまま保存できないため、escapeメソッドを使って文字コードに変換する必要があります。

③変数tmpに「expires = Mon,31-Dec-2015 23:59:59;」を付け足します。その結果、変数tmpには「keyname = (文字コードに変換した)val;expires = Mon,31-Dec-2015 23:59:59;」が代入されています。

※expiresはクッキーの有効期限を指定します。省略した場合の有効期限はブラウザ終了時となります。

④クッキーに変数tmpの内容を保存します。

●getCookie関数

クッキーから値を取り出す関数です。

```
① function getCookie(keyname){
②       var tmp = document.cookie + ";";
③       var index1 = tmp.indexOf(keyname, 0);
```

```
④      if(index1 != -1){
⑤              tmp = tmp.substring(index1, tmp.length);
⑥              index2 = tmp.indexOf("=", 0);
⑦              index3 = tmp.indexOf(";", index2 );
⑧              return unescape(tmp.substring(index2 + 1, index3));
⑨      }
⑩      return "";
⑪ }
```

①引数としてkeynameを受け取ります。これはクッキーから値を取り出すときのキー名となります。

②クッキーの値に「;」(セミコロン)を付加したものを変数tmpに代入します。

※クッキーの値が取得できた場合、日本語などは文字コードとなっているため変数tmpの中身はNAME = %u7FD4%u6CF3%u793E;のようになっています。

③変数tmpをindexOfメソッドで検索します。変数tmpに代入されている文字列中に引数で渡されたキー名(keyname)があった場合、変数index1には最初に一致した位置(0~)が代入されます。なかった場合は-1が代入されます。

④index1の値が-1以外のとき⑤~⑧の処理を行います。

⑤変数tmpからキー名以降の文字列を抜き出してtmpに代入し直します。
この処理はキー名に「=」(イコール)が使用されているときに、⑥で行う処理がおかしくなるのを防ぐためです。このサンプルではありえませんが、念のためこの処理を行っています。

⑥⑤で代入し直した変数tmpの「=」(イコール)の位置を検索し、index2に代入します。

⑦⑥と同じように「;」(セミコロン)の位置を検索し、index3に代入します。

⑧⑥と⑦で取得した値を利用することで必要な部分だけと取り出すことができます。取り出した値が文字コードであるときのことを考えunescapeメソッドで文字列に変換し、関数の呼び出し元にこの値を返して処理を終了します。

⑩④でindex1の値が-1だった場合、空文字を返します。

JavaScript

```
var names;
var divElem;

//ページ読み込み時の処理の関数
function loadPage(){
  names = getCookie("NAME"); //クッキーから名前を取得して変数namesに代入
  divElem = document.getElementById("div");

  if(names == ""){ //クッキーから取得できなかった場合
    divElem.innerHTML = "<p>お名前は？<input type='text' id='name'/></p>"+
                        "<p><input type='button' value='OK' onclick='clickOK()'/></p>";
  }else{ //取得できた場合
    divElem.innerHTML = "<b>ようこそ" + names + "さん！</b>";
  }
```

```
}

//OKボタンがクリックされたとき呼び出される関数
function clickOK(){
    setCookie("NAME", document.getElementById("name").value);
    loadPage();
}

//クッキーから値を読み込む関数
function getCookie(keyname){

    var tmp = document.cookie + ";";
    var index1 = tmp.indexOf(keyname, 0);

    if(index1 != -1){
        tmp = tmp.substring(index1, tmp.length);
        var index2 = tmp.indexOf("=", 0);
        var index3 = tmp.indexOf(";", index2);
        return unescape(tmp.substring(index2 + 1, index3));
    }
    return "";
}

//クッキーに値を書き込む関数
function setCookie(keyname, val){
    var tmp = keyname + "=" + escape(val) + ";";
    tmp += "expires = Mon,31-Dec-2015 23:59:59;";
    document.cookie = tmp;
}
```

HTML

```
<body onload="loadPage()">
<div id="div">
</div>
</body>
```

初回アクセス時は名前をたずねます

名前を入力すると、メッセージが表示されます。2回目以降のアクセスでも、同様のメッセージが表示されます

Column　クッキーとは何か?

クッキーとはユーザー情報やアクセス履歴などの情報をWebサーバーとWebブラウザでやり取りするための仕組み、もしくはその情報が格納されているデータファイルのことです。サーバーからブラウザに送られた情報は、テキスト形式でユーザーのコンピュータに一時的に保存され、次回の交信時に送り返されます。

クッキーにはブラウザの種類、最後にサイトを訪れた日時、そのサイトへの訪問回数、IDやパスワードなどさまざまな情報を書き込むことができます。目的はユーザーの識別で、その一例としてユーザー認証の簡素化、アクセス履歴や商品の購入履歴の把握等に利用されています。Webサイトにアクセスした際、入力欄に前回使用したユーザー名が記録されて次回の入力が省略できたりするのは、この機能を利用しているからなのです。また、クッキーには有効期限を設定でき、有効期限を過ぎたものは消滅します。

そもそもクッキーはNetscape Communications社が開発し、Netscape Persistent Cookiesとして同社のブラウザに組み込んだのが始まりです。標準化団体によって正式に規格化されたものではありませんが、多くのブラウザが対応しているため事実上の世界標準になっています。

cookie プロパティ ・・・・・・・・・・・・・・・・・・・・ P.064

WINDOW.01

新しいウィンドウを開きたい

★ = ▼.open(◆, ▲, ●)

★……新しいウィンドウのオブジェクト
▼……親Windowオブジェクト(ウィンドウ名)【省略可】
◆……URI
▲……新しいウィンドウの名前【省略可】
●……オプション値【省略可】

形式 メソッド

新しいウィンドウを開くメソッドです。タブブラウザの場合は新しいウィンドウではなく、新しいタブを開く場合もあります。新しいウィンドウに表示するHTML/XHTMLファイルのURIは◆で指定します。絶対URIと相対URIいずれの形でも指定可能です。URIを指定しない場合は空白の状態でウィンドウが開きます。

▲はウィンドウの名前です。既に開いているウィンドウやフレームの名前を指定すれば、指定した名前を持つウィンドウに新しいページを表示することも可能です。

新しく開くウィンドウのサイズ、ツールバーやステータスバーの表示の有無などは、オプション値(●)として設定します。オプション値の並び順は問いません。それぞれを「,」(カンマ)で区切り、全体を「" "」(ダブルクォーテーション)でくくって指定します。ブラウザによってはオプションの初期値が異なる場合がありますので、設定が必要なときは記述しておいた方がよいでしょう。また、ブラウザやバージョンによっては有効にならないオプションがあるので注意が必要です。なお、オプション値は以下の通りです。

openメソッドのオプション値	設定値	動作
directories	yes/noまたは1/0	ディレクトリバー表示/非表示
location	yes/noまたは1/0	ロケーションバー表示/非表示
menubar	yes/noまたは1/0	メニューバー表示/非表示
scrollbars	yes/noまたは1/0	スクロールバー表示/非表示
status	yes/noまたは1/0	ステータスバー表示/非表示
toolbar	yes/noまたは1/0	ツールバー表示/非表示
resizable	yes/noまたは1/0	ウィンドウサイズ変更可/変更不可
width	数値(ピクセル単位)	ウィンドウの幅
height	数値(ピクセル単位)	ウィンドウの高さ

文例

```
window.open("", "new", "width=480,height=360,status=1,menubar=1");
```

ステータスバーとメニューバーのみが表示された幅480ピクセル、高さ360ピクセルのnewという名前の空白のウィンドウを開きます。

Column　Windowオブジェクト

WindowオブジェクトはJavaScriptのオブジェクト階層の最上位に位置するオブジェクトで、Webブラウザのウィンドウに関する情報を扱います。新しいウィンドウを開いたり、表示中のウィンドウの情報を取得したり、ウィンドウのサイズや位置などを変更可能です。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

参照

ウィンドウを閉じたい…………………… P.076
別のウィンドウを操作したい…………… P.077
ウィンドウの位置を指定したい………… P.078
【SAMPLE】別のウィンドウを操作する…… P.085

WINDOW.02

ウィンドウを閉じたい

★.close()

★……Windowオブジェクト(ウィンドウ名)【省略可】

形式 メソッド

ウィンドウを閉じるには、closeメソッドを使用します。★に指定したウィンドウを閉じ、ウィンドウ名を省略した場合は自分自身のウィンドウを閉じます。

文例

```
newWin.close();
```

ウィンドウnewWin を閉じます。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照
新しいウィンドウを開きたい……P.074
別のウィンドウを操作したい……P.077
ウィンドウの位置を指定したい……P.078
【SAMPLE】別のウィンドウを操作する……P.085

WINDOW.03

別のウィンドウを操作したい

★.opener	オープン元のウィンドウを参照
★.closed	ウィンドウが閉じているかを参照
★.name	ウィンドウ名を参照／設定

★……親Windowオブジェクト（ウィンドウ名）

形式 プロパティ

親ウィンドウとサブウィンドウとの間で相手のウィンドウを参照するにはopener、closedプロパティを使用します。

openerプロパティ

★にウィンドウ名を指定した場合は指定したウィンドウを開いた親ウィンドウ、ウィンドウ名を省略した場合は自分自身のウィンドウを開いた親ウィンドウを参照します。

closedプロパティ

指定したウィンドウが閉じているかどうかを参照します。閉じている場合はtrue、閉じていない場合はfalseを返します。開いたウィンドウがユーザーによって閉じられてしまったかどうかを調べることができます。

nameプロパティ

ウィンドウ★の名前を参照／設定します。

文例

```
parentLoc = window.opener.location;
```

親ウィンドウのリンク先を変数parentLocに代入します。

```
if(newWin.closed)
    alert("閉じています");
```

ウィンドウnewWin が閉じていた場合、「閉じています」というダイアログを表示します。

```
wName = newWin.name;
```

ウィンドウnewWin の名前を変数wNameに代入します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

参照
新しいウィンドウを開きたい・・・・・・・・・・・・P.074
ウィンドウを閉じたい・・・・・・・・・・・・・・・・P.076
ウィンドウの位置を指定したい・・・・・・・・・・P.078
【SAMPLE】別のウィンドウを操作する・・・・・・P.085

WINDOW.04

ウィンドウの位置を指定したい

★.moveTo(◆, ◇)　ウィンドウ位置を設定
★.moveBy(▲, △)　ウィンドウ位置を相対的に変更

★……Windowオブジェクト(ウィンドウ名)【省略可】
◆……左上端のX座標
◇……左上端のY座標
▲……X方向の変更量
△……Y方向の変更量

形式 メソッド

ウィンドウを移動させるメソッドです。座標や変更量はピクセル単位で指定します。多くの最新のブラウザでは、JavaScriptで開いたわけではないウィンドウや、複数のタブを持つウィンドウでは無視されます。

moveToメソッド

(◆, ◇)で指定したモニタ上の座標位置にウィンドウを移動させます。

moveByメソッド

現在の位置から(▲, △)で指定した分だけウィンドウを移動させます。マイナスの値を指定して、逆方向に移動させることも可能です。

文例

```
subWin.moveTo(100, 50);
```

ウィンドウsubWinをモニタ上の座標(100, 50)に移動させます。

```
window.moveBy(20, 10);
```

現在のウィンドウをX方向に20、Y方向に10ピクセル移動させます。

▶ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照

新しいウィンドウを開きたい……P.074
ウィンドウを閉じたい……P.076
別のウィンドウを操作したい……P.077
ウィンドウのサイズを変更したい……P.079
ウィンドウのサイズを調べたい……P.080
モニタの有効領域を参照したい……P.098
【SAMPLE】ウィンドウの位置とサイズを指定する……P.089

WINDOW.05

ウィンドウのサイズを変更したい

★.resizeTo(◆, ◇)　ウィンドウサイズを変更
★.resizeBy(▲, △)　ウィンドウサイズを相対的に[illegible]

★……Windowオブジェクト（ウィンドウ名）【省略可】
◆……幅（ピクセル）
◇……高さ（ピクセル）
▲……X方向の[illegible]
△……Y方向の[illegible]

[illegible]式　メソッド

ウィンドウの大きさを指定したサイズに変更するメソッドです。サイズや[illegible]はピクセル単位で指定します。ウィンドウのサイズはブラウザの種類によって各種バーやウィンドウの枠を含む／含まないといった違いがあります。なお、多くの[illegible]のブラウザでは、JavaScriptで開いたわけではないウィンドウや、複数のタブを持つウィンドウでは無視されます。

resizeToメソッド

ウィンドウのサイズを（◆, ◇）で指定した幅と高さに変更します。

resizeByメソッド

現在のウィンドウのサイズに（▲, △）で指定した幅と高さを追加し、サイズ変更します。マイナスの値を指定すれば、ウィンドウのサイズを小さくできます。

文例

```
subWin.resizeTo(400, 400);
```

ウィンドウsubWinを幅400、高さ400ピクセルにサイズ変更します。

```
window.resizeBy(50, 100);
```

現在のウィンドウのサイズに[illegible]50、高さ100ピクセルを追加し、サイズ変更します。

▶ ブラウ[illegible]対応表	IE10	[illegible]	[illegible]	Fx	[illegible]	[illegible]	[illegible]	[illegible]	Android
	○	○	○	○	×	○	×	×	×

参照
ウィンドウの位置を指定したい ………… P.078
ウィンドウのサイズを調べたい ………… P.080
モニタの有効領域を参照したい ………… P.098
【SAMPLE】ウィンドウの位置とサイズを指定する‥ P.089

WINDOW.06

ウィンドウのサイズを調べたい

★.innerWidth	ウィンドウの内[illegible]の[illegible]を参照／設定
★.innerHeight	ウィンドウの内側の高さを参照／設定
★.outerWidth	ウィンドウの外側の[illegible]を参照／設定
★.outerHeight	ウィンドウの外側の高さを[illegible]／設定

★……Windowオブジェクト(ウィンドウ名)【省略可】

形式 プロパティ

指定したウィンドウの[illegible]や高さを参照／設定するプロパティです。サイズや[illegible]はピクセル単位で指定します。これらのプロパティの働きは、resizeToやresizeByメソッドに似ていますが、値を参照できる点が異なります。ただし、IE8までのInternet Explorerは未対応です。

innerWidth、innerHeightプロパティ

ウィンドウの内側(表示領域)の幅や高さを参照／設定します。

outerWidth、outerHeightプロパティ

ウィンドウの外側(ウィンドウ全体)の高さや幅を参照／設定します。

文例

```
iw = innerWidth;
```

変数iwに現在のウィンドウの内側の幅を代入します。

```
ih = innerHeight;
```

変数ihに現在のウィンドウの内側の高さを代入します。

```
document.write("<em>ウィンドウの外側の幅：" + outerWidth + "</em>");
```

ウィンドウの外側の幅を表示します。

```
outerHeight = 480;
```

ウィンドウの外側の高さを480ピクセルに設定します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	[illegible]	[illegible]	iOS6	Android
	○	○	×	○	○	○	○	○	○

参照

ウィンドウの位置を指定したい ………… P.078
ウィンドウのサイズを変更したい ………… P.079
モニタの有効領域を参照したい ………… P.098
モニタの表示サイズを参照したい ………… P.100
サイズ[illegible]時に処理を行いたい ………… P.131
【SAMPLE】ウィンドウの位置とサイズを指定する ‥ P.089

WINDOW.07

ページをスクロールさせたい

★.scroll(◆, ◇)	指定位置にスクロールする
★.scrollTo(◆, ◇)	指定位置にスクロールする
★.scrollBy(▲, △)	相対位置にスクロールする

★……Windowオブジェクト（ウィンドウ名）【省略可】
◆……X座標
◇……Y座標
▲……X方向の移動量（ピクセル）
△……Y方向の移動量（ピクセル）

形式 メソッド

ページの内容をスクロールさせるメソッドです。座標や移動量はピクセル単位で指定します。

scroll、scrollToメソッド

(◆, ◇)で指定した絶対位置にスクロールし、ページの内容を表示します。

scrollByメソッド

現在の表示開始位置から(▲, △)で指定した分だけ開始位置を移動させます。マイナスの値を指定して、逆方向に移動させることも可能です。ただし、ページのサイズを超える値を指定した場合は無効になるので、スクロールバーで移動できる範囲内を指定しなければなりません。なお、ページのサイズを調べるにはinnerWidthとinnerHeightプロパティを利用します。

文例

```
scroll(x, y);
```
変数x、yの値の位置にスクロールさせます。

```
newWin.scrollTo(50, 100);
```
ウィンドウnewWinを座標(50,100)にスクロールさせます。

```
newWin.scrollBy(50, 100);
```
ウィンドウnewWinをX方向に50ピクセル、Y方向に100ピクセルスクロールさせます。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照
ドキュメントの端からの位置を参照／設定したい・・・P.082

WINDOW.08

ドキュメントの端からの位置を参照／設定したい

★.pageXOffset 横方向のオフセットを参照／設定

★.pageYOffset 縦方向のオフセットを参照／設定

★……Windowオブジェクト（ウィンドウ名）【省略可】

形式 プロパティ

pageXOffset、pageYOffsetプロパティはそれぞれX方向、Y方向のオフセット、つまり現在の表示開始位置を参照／設定します。これらのプロパティを使用すれば、スクロールされている位置を調べることができます。

文例

```
newWin.pageXOffset = 250;
```

ウィンドウnewWinのX方向のオフセットを250ピクセルに設定する。

```
yo = newWin.pageYOffset;
```

ウィンドウnewWinのY方向のオフセットを変数yoに代入する。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

ページをスクロールさせたい・・・・・・・・・・・・・P.081

WINDOW.09

ブラウザのボタンと同様の処理をしたい

★.back()	1つ前のページに戻る
★.forward()	1つ先のページに進む
★.home()	ホームページに[illegible]
★.print()	印刷する
★.stop()	読み込みを中止
★.find(◆)	指定した文字列を[illegible]

★……Windowオブジェクト（ウィンドウ名）【省略可】
◆……文字列

形式 メソッド

ブラウザのツールバーにあるボタンと同じ働きをするメソッドです。back、forwardメソッドはそれぞれ対象となる履歴がない場合は何も起こりません。printメソッドでは一般的には印刷ダイアログを開きます。

文例

window.back();	1つ前のページに戻ります。
window.forward();	1つ先のページに進みます。
opener.home();	親ウィンドウにホームページを表示します。
myPage.print();	ウィンドウmyPage を印刷します。
onClick="window.stop() "	クリックしたら読み込みを中止します。
find("プロパティ");	ドキュメント内から「プロパティ」という文字列を検索してその部分をブラウザに表示します。

▶ ブラウザ対応表	IE10	[illegible]	[illegible]	Fx	Chrome	Safari	[illegible]	[illegible]	Android
print	○	○	○	○	○	○	○	×	×
find	×	×	×	○	×	×	×	×	×
その他	×	×	×	○	×	×	○	×	×

参照
履歴の前後に移動したい …………………… P.240
履[illegible]の数を調べたい …………………… P.239
【SAMPLE】ブラウザのボタンと[illegible]処理をしたい … P.083

インラインフレームを参照したい

★.contentDocument	インラインフレーム内のDocumentオブジェクト(読取専用)
★.contentWindow	インラインフレーム内のWindowオブジェクト(読取専用)
★.src	インラインフレーム内に表示する内容のURL
★.srcdoc	インラインフレーム内に表示する内容のHTMLコード
★.sandbox	インラインフレームのサンドボックス設定(読取専用)
◆.frames[参照番号]	フレームもしくはインラインフレームを格納した配列

★……IFrameオブジェクト(インラインフレーム名)
◆……Windowオブジェクト(ウィンドウ名)

形式 プロパティ

HTML5ではアクセシビリティの問題からフレームは廃止となり、代わりにインラインフレーム(iframe要素)が標準に昇格しました。インラインフレームを使うと、ドキュメント内に別のドキュメントを埋め込むことができます。

sandboxプロパティが存在する場合、埋め込まれた側のドキュメントは「サンドボックス(砂場)化」されており、埋め込んでいる親側へのアクセス、ウィンドウ操作、スクリプトの実行、データの送信などの危険な操作が行えません(空文字が指定されている場合)。空文字以外のキーワードが指定されている場合は、指定に基づいて制限が緩和されます(サンプル参照)。

文例

```
var insideWin = iframe.contentWindow;
```
インラインフレームiframe内のWindowオブジェクトを取得する。

```
var isSandBox = iframe.hasAttribute("sandbox");
```
インラインフレームiframeにサンドボックス指定がされているかどうかを取得する。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

【SAMPLE】インラインフレームを操作する ‥P.084

WINDOW.SAMPLE-01

別のウィンドウを操作する

オープン元のウィンドウから新しく開いたウィンドウの背景色を操作するサンプルです。元のウィンドウが立ち上がると、新たにサブウィンドウが開き、元のウィンドウのセレクトボックスで色を選択することによってサブウィンドウの背景色を変更できます。元のウィンドウの[サブウィンドウを閉じる]ボタンではサブウィンドウを閉じることができます。

存在しないウィンドウ(既に閉じているウィンドウ)を操作しようとすると、エラーになるので各操作を実行する前にウィンドウが閉じているか確認しましょう。閉じている場合はreturnで関数から抜けて処理を行わないようにして、ダイアログを表示します。

JavaScript

```
var newWin;

//新しいウィンドウを開く関数
function openWin(){
    window.name = "openerWin"; //オープン元ウィンドウの名前を"openerWin"に設定
    newWin = window.open("subwindow.html", "newWin", "status=1,width=480,
height=240");//変数newWinは他の関数でも使用するため、var宣言はしない
}

//サブウィンドウを閉じる関数
function closeWin(){

    //サブウィンドウが既に閉じている場合、ダイアログを表示
    if(newWin.closed){
        alert("既に閉じています");
        return;
    }

    newWin.close();
    alert("サブウィンドウを閉じました");
}

//オープン元ウィンドウの名前を参照する関数
function getName(){
    document.getElementById("text1").value = window.opener.name;
}

//サブウィンドウの背景色を変更する関数
```

```
function changeColor(index){

    //サブウィンドウが既に閉じている場合、ダイアログを表示
    if(newWin.closed){
        alert("サブウィンドウが既に閉じているので操作できません。");
        return;
    }

    //セレクトボックスのindexの値を参照して背景色を変更
    switch(index){
        case 0:
            break;
        case 1:
            newWin.document.bgColor = "#ffffff";
            break;
        case 2:
            newWin.document.bgColor = "#ff0000";
            break;
        case 3:
            newWin.document.bgColor = "#00ff00";
            break;
        case 4:
            newWin.document.bgColor = "#0000ff";
            break;
    }
}
```

HTML

```
<body onload="openWin()">
    <form action="" >
    <p>
        <b>サブウィンドウの背景色を変更します</b>
        <select onchange="changeColor(selectedIndex)">
            <option selected="selected">選択してください</option>
            <option>白</option>
            <option>赤</option>
            <option>緑</option>
            <option>青</option>
        </select>
    </p>
    <p><input type="button" value="サブウィンドウを閉じる" onclick="closeWin()"
/></p>
    </form>
</body>
```

HTML

※subwindow.html

```
<body onload="getName()">
  <form action="">
    <p>元ウィンドウの名前  :<input type="text" id="text1" /></p>
  </form>
</body>
```

Internet Explorer

❶サブウィンドウ(subwindow.html)が表示されます。サブウィンドウにはgetName関数によって親ウィンドウ名が表示されます

❷[サブウィンドウを閉じる]ボタンをクリックするとサブウィンドウが閉じます。サブウィンドウを閉じた状態で、[サブウィンドウを閉じる]ボタンをクリックすると、警告ダイアログが表示されます

❸元のウィンドウではサブウィンドウの背景色を変更することが可能です

参照
open プロパティ ………… P.074
close プロパティ ………… P.076
opener プロパティ ………… P.077
closed プロパティ ………… P.077
name プロパティ ………… P.077

ウィンドウの位置とサイズを指定する

ウィンドウのサイズを指定し、モニタの中央に表示するサンプルです。ウィンドウをモニタの中央に表示するには、まずモニタのサイズを調べ、その値を2で割った値を取得します。さらに表示するウィンドウの幅、高さの1／2を引いた座標をmovoToメソッドで指定すれば実現できます。

ウィンドウ内のドキュメントはウィンドウの各サイズを取得して表示します。その際、ブラウザによってプロパティへの対応が異なるため、処理を振り分けています。

なお、Internet Explorerは、IE8までは、outerWidth、outerHeight、innerWidth、innerHeightプロパティに対応していません。innerWidth、innerHeightプロパティの代わりにclientWidth、clientHeightプロパティを使用してウィンドウの内側の幅と高さを取得できます。ただし、clientWidth、clientHeightプロパティは表示モードによって参照方法が異なるため、document.compatModeプロパティで表示モードを取得します。※準準拠モードの場合はCSS1Compat、互換モードの場合はBackCompatが返ります。標準準拠モードの場合はdocument.documentElement.clientWidth、互換モードの場合はdocument.body.clientWidthで取得します。また、このサンプルでは「ウィンドウの横幅」(outerWidth)、「ウィンドウの縦幅」(outerHeight)には「参照不可」と表示します。

※ブラウザでは表示モードをDOCTYPE宣言で判断しています。本サンプルは、各ブラウザの最新バージョンでは標準モードとして認識されます。

JavaScript

```
resizeTo(400,400); //ウィンドウサイズを設定する
moveTo(screen.width/2-200,screen.height/2-200); //ウィンドウを中央に表示させる
window.onresize = resizeWin; //ウィンドウのサイズが変更されたとき、resizeWin関数を呼び出す

//ウィンドウのサイズを参照する関数
function resizeWin(){
    var ieStr = "参照不可";
    var elementId = document.getElementById("form1"); //繰り返し使う宣言等は変数に代入する

    //ブラウザによって振り分け
    //innerWidthが有効なブラウザ(Internet Explorer以外)
    if(window.innerWidth){
        elementId.text[0].value = document.compatMode;
        elementId.text[1].value = window.innerWidth;
        elementId.text[2].value = window.innerHeight;
```

```
        elementId.text[3].value = window.outerWidth;
        elementId.text[4].value = window.outerHeight;
    //Internet Explorer 標準準拠モード
    }else if(document.documentElement && document.documentElement.
clientWidth != 0){
        elementId.text[0].value = document.compatMode;
        elementId.text[1].value = document.documentElement.clientWidth;
        elementId.text[2].value = document.documentElement.clientHeight;
        elementId.text[3].value = ieStr;
        elementId.text[4].value = ieStr;
    //Internet Explorer 互換モード
    }else{
        elementId.text[0].value = document.compatMode;
        elementId.text[1].value = document.body.clientWidth;
        elementId.text[2].value = document.body.clientHeight;
        elementId.text[3].value = ieStr;
        elementId.text[4].value = ieStr;
    }
}
```

HTML

```
<body onload="resizeWin()" >
    <form action="" id="form1">
        <p><b>ブラウザの表示モード</b><input type="text" name="text" /></p>
        <p><b>ウインドウ内側の横幅</b><input type="text" name="text" /></p>
        <p><b>ウインドウ内側の縦幅</b><input type="text" name="text" /></p>
        <p><b>ウインドウ外側の横幅</b><input type="text" name="text" /></p>
        <p><b>ウインドウ外側の縦幅</b><input type="text" name="text" /></p>
    </form>
</body>
```

Internet Explorer 8

IE8はouterWidth、outerHeightプロパティに対応していないため、「ウィンドウ外側の横幅」、「ウィンドウ外側の縦幅」には「参照不可」と表示されます

Firefox

モニタの中央にウィンドウが表示されます
ウィンドウのサイズを変更すると、各サイズの数値が更新されます

参照

moveTo メソッド ･････････････････ P.078
resizeTo メソッド ･････････････････ P.079
innerWidth プロパティ ･･･････････････ P.080
innerHeight プロパティ ･･･････････････ P.080
outerWidth プロパティ ･･･････････････ P.080
outerHeight プロパティ ･･･････････････ P.080

WINDOW.SAMPLE-03

ブラウザのボタンと同様の処理をする

ブラウザのボタンやメニュー項目と同じ働きをするボタンをウィンドウ上に配置したサンプルです。Internet Explorerではprintメソッド以外は対応していないため、処理を分岐させてダイアログを表示するようにしています。

JavaScript

```
var ieStr = "Internet Explorerではこの操作はできません。";
                                    ダイアログに表示するメッセージ
var boolIE = navigator.appName.indexOf("Microsoft")>-1;
                          ブラウザがInternet Explorerかどうかのbool値

//戻る処理の関数
function backPage(){
    if(boolIE){
        alert(ieStr);
        return;
    }
    back();
}

//進む処理の関数
function forwardPage(){
    if(boolIE){
        alert(ieStr);
        return;
    }
    forward();
}

//ホームに移動する処理の関数
function homePage(){
    if(boolIE){
        alert(ieStr);
        return;
    }
    home();
}

//読み込みを中止する処理の関数
```

```
function stopPage(){
    if(boolIE){
        alert(ieStr);
        return;
    }
    stop();
}

//検索する処理の関数
function findPage(){
    if(boolIE){
        alert(ieStr);
        return;
    }
    find();
}
```

HTML

```
<body>
    <form action="">
        <p>
        <input type="button" value="戻る" onclick="backPage()" />
        <input type="button" value="進む" onclick="forwardPage()" />
        <input type="button" value="ホーム" onclick="homePage()" />
        <input type="button" value="印刷" onclick="print()" />
        <input type="button" value="中止" onclick="stopPage()" />
        <input type="button" value="検索" onclick="findPage()" />
        </p>
    </form>
</body>
```

Internet Explorer

[印刷]ボタンはブラウザの[印刷]ボタンと同様の効果を実現します。
ただし、それ以外のボタンはInternet Explorerでは動作しません

Firefox

Firefoxでは[検索]ボタンを含めすべてのボタンが使用できます

参照
back メソッド ･･････････････････････ P.083
forward メソッド ････････････････････ P.083
home メソッド ･･････････････････････ P.083
print メソッド ･･････････････････････ P.083
stop メソッド ･･････････････････････ P.083
find メソッド ･･････････････････････ P.083

WINDOW.SAMPLE-04

インラインフレームを操作する

インラインフレームで、包含する「親」と包含される「子」の、2つのドキュメントの間の、スクリプトによる相互アクセスを実装したサンプルです。親から子のドキュメントはiframeオブジェクトのcontentDocumentプロパティで取得し、子から親のドキュメントはwindow.parent.documentで取得します。

sandbox属性に「allow-scripts allow-same-origin」の2つのキーワードを指定することで、サンドボックス環境でありつつ、子ドキュメントにスクリプト実行と親ドキュメントへのアクセスを許可しています。allow-same-originが指定されないと、二つのドキュメントは強制的に別ドメインの扱いとなり、基本的には親から子へのアクセスもできません。

キーワードには以下のようなものがあります。複数のキーワードは半角スペースで区切って指定します。

キーワード	説明
allow-scripts	JavaScriptの実行を許可します。
allow-forms	フォームの送信を許可します。
allow-same-origin	強制的な別ドメイン扱いによるアクセス制限を解除します。
allow-top-navigation	トップ ウィンドウの操作を許可します。
allow-popups	ウィンドウを開く操作を許可します。

JavaScript

※親(外)側のドキュメントのJavaScript

```
function onLoadHandler(){
    var iframe = document.getElementById("myFrame");
    var inside_doc = iframe.contentDocument;
    var inside_div = inside_doc.getElementById("main_area02");
    inside_div.innerHTML = "この文章は親ドキュメントから<br>内側のドキュメントへの書
き込みです。";
}
```

HTML

※親(外)側のドキュメントのHTML

```
<body>
<iframe
   sandbox="allow-scripts allow-same-origin"
   onload="onLoadHandler()"
   id="myFrame"
   src="window04_inside_4th.html">
</iframe>

<div id="main_area01"></div>
</body>
```

JavaScript

※子(内)側のドキュメントのJavaScript

```
var outside_doc = window.parent.document;
var outside_div  = outside_doc.getElementById("main_area01");
outside_div.innerHTML = "この文章は内側のドキュメントから<br>親ドキュメントへの書き
込みです。";
```

HTML

※子(内)側のドキュメントのHTML

```
<body>
<div id="main_area02"></div>
</body>
```

インラインフレームの中身に外側から文字列を書き込み、同時に、内側からも外側のドキュメントに文字列を書き込んでいます

contentDocument プロパティ・・・・・・・・・・・ P.084

SCREEN.01

モニタの有効領域を参照したい

★.availHeight	有効な領域の高さを参照
★.availWidth	有効な領域の幅を参照
★.availLeft	有効な左端のX座標を参照
★.availTop	有効な上端のY座標を参照

★……Screenオブジェクト

形式 プロパティ

モニタの有効領域のサイズ、または端の座標を参照します。

availWidth、availHeightプロパティ

モニタの表示サイズからメニューバーやタスクバーなどの部分を除いた有効領域のサイズを参照するプロパティです。availWidthプロパティは有効領域の幅、availHeightプロパティは有効領域の高さを返します。

availLeft、availTopプロパティ

モニタにおける有効領域の端の座標を参照します。availLeftプロパティは有効な左端のX座標、availTopプロパティは有効な上端のY座標を返します。

文例

```
aw = screen.availWidth;
```
有効な領域の幅を変数awに代入します。

```
ah = screen.availHeight / 2;
```
有効な領域の高さを2で割った値を変数ahに代入します。

```
document.write("表示サイズの■" + screen.availLeft);
```
モニタの有効なX座標を書き出します。

```
at = screen.availTop;
```
モニタの有効なY座標を変数atに代入します。

Column　Screenオブジェクト

Screenオブジェクトはモニタの表示サイズや使用できる色の数（カラーパレット）の設定など画面に関する[illegible]を参照／設定できます。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	△	△	△	○	○	○	△	○	○

※ IE と Opera は availLeft、availTop に非対応

ウィンドウの位置を指定したい ･････････････ P.078
ウィンドウのサイズを変更したい･･･････････ P.079
ウィンドウのサイズを調べたい ････････････ P.080

SCREEN.02

モニタの表示サイズを参照したい

★.width　　モニタの高さを参照
★.height　　モニタの幅を参照

★……Screenオブジェクト

形式　プロパティ

モニタの表示サイズを参照します。widthプロパティは幅、heightプロパティは高さを表します。

文例

```
w = screen.width;
```
モニタの表示サイズ(幅)を変数wに代入します。

```
alert("このモニタの表示サイズの高さは" + screen.height + "です。");
```
モニタの表示サイズ(高さ)をダイアログに表示します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照
ウィンドウの位置を指定したい ………… P.078
ウィンドウのサイズを変更したい ………… P.079
ウィンドウのサイズを調べたい ………… P.080

SCREEN.03

モニタの表示色の設定を参照したい

★.colorDepth　表示できる色数を参照
★.pixelDepth　オフスクリーンの色深度を参照

★……Screenオブジェクト

形式 プロパティ

モニタの表示色の設定を参照します。

colorDepthプロパティ

モニタの色深度（表示できる色数）をビット値で返します。

pixelDepthプロパティ

オフスクリーンの色深度（1ピクセルに必要なビット数）をビット値で返します。返される値は右の表の通りです。なお、IE8までのInternet ExplorerはpixelDepthプロパティに対応しておらず、値として常にundefinedが返されます。

色数	値
白黒	1
16色	4
256色	8
65,536色（HighColor）	16
1,677万色（TrueColor）	32

文例

```
if(screen.colorDepth <= 16) {
    alert("TrueColorの環境でご覧ください。");
}
```

HighColor 以下の場合、「TrueColor の環境でご覧ください。」というダイアログを表示します。

```
alert(screen.pixelDepth);
```

オフスクリーンの色深度をダイアログに表示します。

ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	△	○	○	○	○	○	○

※ IE8 は pixcelDepth に非対応

SCREEN.SAMPLE-01

ウィンドウを画面中央に表示する

ウィンドウの幅と高さをモニタの有効範囲の半分のサイズにし、このウィンドウをモニタの有効範囲の中央に表示するサンプルです。

ウィンドウの幅と高さをモニタの有効範囲の半分のサイズにするためには、まずavailWidth、availHeightプロパティでモニタの有効範囲のサイズを取得し、その値を2で割った値をresizeToメソッドの引数に指定してサイズを変更します。

次にmoveToメソッドを使用して、ウィンドウの表示位置を設定します。moveToメソッドは引数に指定したモニタ左上からの座標位置にウィンドウを移動させるメソッドです。そのため、幅、高さともに半分にリサイズしたウィンドウをモニタ中央に表示するには、ウィンドウの左上の座標がモニタの有効範囲の1/4の位置になるようにします。

JavaScript

```
resizeTo(screen.availWidth/2, screen.availHeight/2); //ウィンドウサイズの設定
moveTo(screen.availWidth/4, screen.availHeight/4); //ウィンドウ表示位置の設定
```

HTML

```
<body>
<p><b>このウィンドウは画面の中央に表示されます</b></p>
</body>
```

Internet Explorer

画面がモニタ中央に表示されます

参照

availHeight プロパティ ・・・・・・・・・・・・・・・ P.098
availWidth プロパティ ・・・・・・・・・・・・・・・ P.098

FORM.01

フォームを参照したい

★.フォーム名	フォームを参照
forms[参照番号]	フォームを参照
★.forms.length	フォームの数を参照
◆.name	フォームの名前を参照

★……Documentオブジェクト
◆……Formオブジェクト(フォーム名またはforms[参照番号])

形式 オブジェクト(Form)、プロパティ(forms.length、name)

Formオブジェクト

フォームを参照するには、名前(フォーム名)や参照番号を利用します。フォーム名は<form>タグのname属性あるいはid属性で指定されているフォームの名前です。たとえば、<form name="enqForm">と指定されているフォームを参照する場合は次のようになります。

```
document.enqForm
document.forms["enqForm"]
```

参照番号を利用する方法は、ドキュメント中のフォームを要素とする配列から、その参照番号に対応するフォームにアクセスします。指定する際はforms[参照番号]という形式になります。フォームの参照番号はドキュメントの中にフォームが記述されている順番で、0からの連番になります。たとえば、ドキュメント中の2番目のフォームを参照する場合は次のようになります。

```
document.forms[1]
```

forms.lengthプロパティ

ドキュメントの中にあるすべてのフォームの数を参照します。

nameプロパティ

フォームの名前を参照します。

文例

```
fName = document.forms[1].name;
```

2番目のフォームの名前を変数fNameに代入します。

```
alert("このドキュメントに含まれるフォームの数：" + document.forms.length);
```

ドキュメントに含まれるフォームの総数をダイアログに表示します。

Column　フォームをDOMで参照する場合

<form id ="enqForm">と指定されているフォームをDOMで参照する場合は次のようになります。

```
document.getElementById("enqForm")
```

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

フォームの部品を参照したい・・・・・・・・・・・・・P.108

FORM.02

フォームの送信先や送信方法を設定したい

★.action	送信先を参照／設定
★.encoding	エンコード方式を参照／設定
★.method	送信形式を参照／設定
★.target	ターゲット名を参照／設定

★……Formオブジェクト（フォーム名またはforms[参照番号]）

形式 プロパティ

form要素の各属性で指定されるフォームの送信先や送信方法に関するプロパティです。これらの設定にJavaScriptを利用することにより、ユーザーの選択や環境によって送信先やエンコード方式、送信方法を変更できます。

actionプロパティ

form要素のaction属性に該当するデータの送信先を参照／設定します。

encodingプロパティ

form要素のenctype属性に該当するデータを送信する際のエンコード方式（MIMEタイプ）を参照／設定します。

methodプロパティ

method属性で指定する送信方式を参照／設定します。値はpostまたはgetになります。

targetプロパティ

フォームの内容を送信した結果を表示させるフレーム名またはウィンドウ名を参照／設定します。

文例

```
document.q_form.action = "/cgi-bin/question.cgi";
```
フォーム名q_formの送信先を/cgi-bin/question.cgiにします。

```
document.q_form.encoding = "application/x-www-form-urlencoded";
```
フォーム名q_formのエンコード方式をapplication/x-www-form-urlencodedにします。

```
document.q_form.method = "post";
```
フォーム名q_formの送信形式をpostにします。

```
document.q_form.target = "subWin ";
```
フォーム名q_formのターゲットウィンドウをsubWinにします。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

フォームの内容をリセット／送信したい・・・・・P.110
【SAMPLE】フォームの送信先や送信方法を設定する・・P.120

FORM.03

フォームの部品を参照したい

★.エレメント	エレメントを参照
★.elements[参照番号]	エレメントを参照
★.length	エレメントの数を参照
◆.length	エレメントの数を参照
◆.type	エレメントの種類を参照
◆.name	エレメントの名前を参照

★……Formオブジェクト(フォーム名またはforms[参照番号])
◆……Elementオブジェクト(エレメント名またはelements[参照番号])

オブジェクト(Element)、プロパティ(length、length、type、name)

ボタン、チェックボックス、選択メニューなどのフォームの各部品をエレメントといいます(エレメントの種類については次ページを参照)。

elementsオブジェクト

エレメントを参照するには、エレメントの名前(エレメント名)や参照番号を利用します。エレメント名とは<input>などエレメントを定義する各タグのname属性あるいはid属性で設定されている名前です。たとえば<input type="text" name="mail" / >と指定されているエレメントを参照する場合は次のようなります。

```
★.mail
★.elements["mail"]
```

★……エレメントが含まれるフォーム

参照番号を利用する方法は、フォーム中のエレメントを要素とする配列からその参照番号に対応するエレメントにアクセスします。指定する際はelements[参照番号]という形式になります。エレメントの参照番号は、フォームの中でエレメントが記述されている順番で、0からの連番です。たとえばフォーム中の2番目のエレメントを参照する場合は次のようになります。

```
★.elements[1]
```

★……エレメントが含まれるフォーム

lengthプロパティ

フォームにあるすべてのエレメントの数を参照します。FormオブジェクトのlengthプロパティとElementの配列のlengthプロパティは同じ値を返します。

typeプロパティ

エレメントの種類を参照します。エレメントには以下の種類があります。

エレメント	意味
button	ボタン
checkbox	チェックボックス
fileUpload	ファイルアップロード
hidden	隠しオブジェクト
options	選択メニューの項目
password	パスワード入力欄
radio	ラジオボタン
reset	リセットボタン
select	選択メニュー
submit	送信ボタン
text	1行の入力フィールド
textarea	複数行の入力フィールド

HTML5では、右の表のフィールドが追加されました。

エレメント	意味
email	メール入力フィールド
url	URL入力フィールド
search	検索語句入力フィールド
tel	電話番号入力フィールド
number	数値入力フィールド
range	数値入力スライダー

文例

```
document.write(document.test_form.mail.value);
```

フォーム名test_formのエレメント名mailに入力されている文字列を表示します。

```
document.test_form.elements[0].focus();
```

フォーム名test_formの1番目のエレメントにフォーカスを合わせます。

Column　エレメントをDOMで参照する場合

<input type="text" name="mail" id="mail"/ >と指定されているエレメントをDOMで参照する場合は次のようになります。

```
★.getElementById("enqForm")
★……エレメントが含まれるフォーム
```

ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

フォームを参照したい……P.104
【SAMPLE】選択されている項目を調べる……P.124

FORM.04

フォームの内容をリセット／送信したい

★.reset()　フォームの内容をリセット
★.submit()　フォームの内容を送信

★……Formオブジェクト（フォーム名またはforms[参照番号]）

形式 メソッド

それぞれフォームの内容をリセット、送信するメソッドです。submitメソッドを利用して送信する場合には、セキュリティ警告が表示されたり、期待したとおりに動作しなかったりすることがあるので注意が必要です。

文例

```
document.form1.reset();
```
フォーム名form1の内容をリセットします。

```
document.form1.submit();
```
フォーム名form1の内容を送信します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照

フォームの送信／リセット時に処理を行いたい‥P.136
【SAMPLE】フォームの内容を送信する‥‥‥P.126

FORM.05

選択されているかを調べたい

★.checked　チェック状態を参照／設定
★.options[参照番号].selected　選択状態を参照

★……Elementオブジェクト（エレメント名またはelements[参照番号]）

形式　プロパティ

チェックボックスやラジオボタン、セレクトメニューのメニュー項目の状態を参照／設定するプロパティです。

checkedプロパティ

指定したチェックボックスやラジオボタンの選択状態を参照／設定します。選択されている場合はtrue、されていない場合はfalseを返します。

selectedプロパティ

指定したメニュー項目の選択状態を参照／設定します。選択されている場合はtrue、されていない場合はfalseを返します。メニュー項目はoptions[参照番号]という形式で指定します。1番目のメニュー項目の参照番号は0になります。

文例

```
document.dataForm.elements[0].checked = true;
```

フォーム名dataFormの1番目のエレメント（チェックボックスやラジオボタン）を選択されている状態（オン）にします。

```
if(document.dataForm.dataMenu.options[0].selected == true) {
   alert("確認してください");
}
```

セレクトメニュー名dataMenuの1番目の項目が選択された場合、「確認してください」というダイアログを表示します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

フォームの部品を参照したい ･････････････ P.108
どの項目が選択されているかを調べたい ････ P.112
選択の初期状態を調べたい ･･････････････ P.113

FORM.06

どの項目が選択されているかを調べたい

★.selectedIndex

★……Elementオブジェクト(エレメント名またはelements[参照番号])

式 プロパティ

セレクトメニューで選択されている項目の参照番号を参照します。最初のメニュー項目の参照番号は0になります。

文例

```
listn = document.form1.s01.selectedIndex;
```

フォーム名form1のセレクトメニュー名s01 で選択されている項目の参照番号を変数listnに代入します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照

フォームの部品を参照したい・・・・・・・・・・・・・・P.108
選択されているかどうかを調べたい・・・・・・・・P.111
選択の初期状態を調べたい・・・・・・・・・・・・・・P.113
フォーム操作時に処理を行いたい・・・・・・・・・・P.138
【SAMPLE】フォームの部品を参照する・・・・・・P.122

FORM.07

選択の初期状態を調べたい

◆.defaultChecked 初期チェック状態を参照
◆.options[参照番号].defaultSelected 初期選択状態を参照

◆……Elementオブジェクト（エレメント名またはelements[参照番号]）

形式 プロパティ

チェックボックスやラジオボタン、セレクトメニューのメニュー項目の初期状態を参照／設定するプロパティです。

defaultCheckedプロパティ

チェックボックスやラジオボタンの初期の選択状態を参照します。選択されている場合はtrue、されていない場合はfalseを返します。

defaultSelectedプロパティ

セレクトメニューの初期の選択状態を参照します。選択されている場合はtrue、されていない場合はfalseを返します。

文例

```
n = document.forms[0].elements[1].options[0].defaultSelected;
```

1番目のフォームの2番目のエレメントの1番目の選択項目の初期選択状態を変数nに代入します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

参照

フォームの部品を参照したい……P.108
選択されているかどうかを調べたい……P.111
どの項目が選択されているかを調べたい……P.112
フォームの部品に表示されるテキストを設定したい……P.114

FORM.08

フォームの部品に表示されるテキストを設定したい

★.value エレメントの文字列を参照／設定

★.defaultValue テキストエリアの初期文字列を参照

★.options[参照番号].text 選択メニューの項目を参照／設定

★……Elementオブジェクト（エレメント名またはelements[参照番号]）

形式 プロパティ

入力欄に初期状態で入力されている文字列やボタンに表示する文字列などを参照／設定するプロパティです。

valueプロパティ

入力欄に入力されている文字列やボタンに表示する文字列などの各エレメントの値を参照／設定します。

defaultValueプロパティ

textarea要素（<textarea>タグ）のテキストエリアに初期状態で入力されている文字列を参照します。

textプロパティ

セレクトメニューのoption要素（<option>タグ）で設定される値を参照または設定します。

文例

```
if(document.enqForm1.mail.value == "") alert("メールアドレスを入力してください")
```

エレメント名mailの値が「""」（空）の場合、「メールアドレスを入力してください」とダイアログに表示します。

```
alert(document.enqForm1.comment.defaultValue);
```

エレメント名commentに初期状態で入力されている文字列をダイアログに表示します。

```
document.enqForm1.enqMenu.options[0].text = "quality";
```

セレクトメニュー名enqMenuの1番目の選択項目の値をqualityに設定します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照

フォームの部品を参照したい・・・・・・・・・・・・・・P.108
選択の初期状態を調べたい・・・・・・・・・・・・・・P.113
【SAMPLE】フォームの部品を参照する・・・・・・P.122

FORM.09

自動的にフォーカスを移動させたい

★.click() クリックする
★.blur() フォーカスを外す
★.focus() フォーカスを合わせる
★.select() 文字を選択状態にする

★……Elementオブジェクト（エレメント名またはelements[参照番号]）

形式 メソッド

ボタンのクリック、フォーカスの移動、入力欄の文字列の選択などを自動的に行うメソッドです。selectメソッドで文字を選択するには、あらかじめ対象とするフォームや入力フィールドテキストエリア内にfocusメソッドでフォーカスを合わせておく必要があります。

文例

```
document.form1.btn1.click();
```
フォーム名form1のエレメント名btn1を自動的にクリックします。

```
document.forms [0].elements[0].blur();
```
1番目のフォームの1番目のエレメントのフォーカスを外します。

```
document.form1.nickname.focus();
```
フォーム名form1のエレメント名nicknameにフォーカスを合わせます。

```
document.form1.nickname.select();
```
フォーム名form1のエレメント名nicknameの文字列を選択状態にします。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照
フォームの部品を参照したい ･･････････････ P.108
フォーカスの移動時に処理を行いたい ･･････ P.132
【SAMPLE】フォームの部品を参照する･･････ P.122

FORM.10

数値入力フィールドとスライダーを操作したい

★.max	入力できる最大値
★.min	入力できる最小値
★.step	値を増減させる際の最小単位
★.stepUp(◆)	stepの値×引数◆だけ値を増加させる
★.stepDown(◆)	stepの値×引数◆だけ値を減少させる

★……type属性に"number"また"range"が設定されたinput要素
◆……何ステップ分だけ値を増減させるかを指定する数値。省略時は1。

書式 メソッド

HTML5では入力フォームのinput要素が拡張され、フィールドの種類(type)が追加されました。

type属性に"number"を設定すると、その入力フィールドは数値入力専用になり、グラフィカルな表示を行うブラウザでは、フィールドに数値を上下させるボタンが入力補助として表示されます。また、"range"を設定すると、その入力フィールドはスライダー形式での数値入力フィールドになり、つまみを動かして値を増減できるようになります。

minで入力できる最小値、maxで入力できる最大値、stepで値の増減の最小単位を指定します。スライダーの場合、maxがちょうど右端の位置に相当し、stepの単位でしかスライダーのつまみを動かせなくなります。

数値フィールドの矢印ボタンやスライダーでの値の増減と同じように、メソッドを使っても値を増減できます(value属性への代入も従来と同じく使用可能です)。

stepUp(◆)メソッド

step属性の値×引数◆の分だけ値を増加させます。たとえばstepが2で引数◆が3なら値が6増加します。

stepDown(◆)メソッド

step属性の値×引数◆の分だけ値を減少させます。たとえばstepが2で引数◆が3なら値が6減少します。

文例

```
document.getElementById("myRange").stepUp(10)
```

myRangeというスライダーの値を10ステップ分増加させます。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照

フォームの部品を参照したい・・・・・・・・・・・・P.108
フォーム操作時に処理を行いたい・・・・・・・・・・P.138

FORM.11

入力制限をしたい

★.required	必須チェックを行うかどうか
★.pattern	入力チェックに使用する正規表現
★.validity	入力チェック状態を表すオブジェクトを取得
★.validationMessage	入力チェック結果のメッセージ。読取専用
★.willValidate	入力チェックの対象にするかどうか
★.checkValidity()	入力チェックを明示的に行い、成功ならtrueを返す
★.setCustomValidity(◆)	カスタム入力エラーを設定する

★……input要素
◆……カスタムエラーメッセージ

形式 メソッド

HTML5ではJavaScriptによる複雑なコーディング抜きで入力チェック・入力制限を行うことができるようになりました。

すでに触れた"number"(数値)、"range"(数値スライダー)のほかに、"email"(メールアドレス)、"url"(URL)、"search"(検索語句)、"tel"(電話番号)といったtype属性がinput要素に追加され、それぞれの種別に応じた入力補助と入力制限を行ってくれます。

多くのケースでは適切なtype属性を指定するだけで十分ですが、独自に細かい入力チェックの制御を行うためのプロパティやメソッドも用意されています。

特定の入力フィールドを必須項目にするには、required属性をtrueにします。

text、search、tel、url、emailの各フィールドの入力内容を正規表現によって制限することもできます。その場合には、pattern属性に文字列で正規表現を指定します。このとき、部分ではなくそのフィールドの入力全体に対して正規表現との一致がチェックされます。

各入力フィールドの直近の検証結果はvalidity属性で取得できます。validity属性に格納されているValidityStateオブジェクトは以下の属性を持っていて、入力チェックの結果を知ることができます。エンターキーや送信ボタン押下時だけでなく、任意のタイミングで入力を検

証したいときは、checkValidityメソッドを呼び出します。

属性名	内容
valueMissing	trueならば、入力が空のため、必須チェック違反になっています。
typeMismatch	trueならば、入力内容がフィールドの期待しているtypeと違っています。
patternMismatch	trueならば、設定された値が正規表現と一致しません。
tooLong	trueならば、maxLengthに設定された入力文字数を超過しています。
rangeUnderflow	trueならば、minで指定された下限値を下回っています。
rangeOverflow	trueならば、maxで指定された上限値を上回っています。
stepMismatch	trueならば、stepで指定された単位での増減と値が不一致です。
customError	trueならば、カスタム入力チェックに違反しています。
valid	trueならば、入力チェックは成功しています。

検証失敗時のメッセージはvalidationMessageで取得できます。これは読み取り専用属性です。

特定のフィールドを検証の対象から外すにはwillValidate属性にfalseをセットします。

独自のロジックで入力チェックを行うには、検証失敗時に、独自のエラーメッセージ◆をsetCustomValidity(◆)メソッドを使ってセットします。◆が空文字でなければ、そのフィールドはチェックに失敗したものとみなされ、上記のvalidityメソッドでもcustomErrorがtrueになります。

文例

```
textInput.onchange = function(ev){
    if(textInput.value != "ok"){
        textInput.setCustomValidity("not ok!");
    } else {
        textInput.setCustomValidity("");
    }
}
```

typeがtextのinput要素textInputにカスタムの入力チェックを追加しています。
ここでは値がokではない場合にカスタムエラーとして検証を失敗させています。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照
フォームの部品を参照したい・・・・・・・・・・・・・・P.108
フォーム操作時に処理を行いたい・・・・・・・・・・P.138

FORM.SAMPLE-01

フォームの送信先や送信方法を設定する

　送信先を変更するサンプルです。[送信]ボタンがクリックされたとき(onsubmitイベント)、sendForm関数を呼び出してラジオボタンのチェックされている項目にしたがってactionプロパティを変更し、送信先を変更しています。また、ラジオボタンのチェックが[感想]だった場合のみtargetプロパティでblankを設定して別ウィンドウで表示させます。

JavaScript

```
//フォームの送信先を変更する関数
function sendForm(){
    var formElem = document.getElementById("form1");
    formElem.method = "get"; //送信形式を設定
    formElem.encoding = "application/x-www-form-urlencoded";
                                                    //エンコード方式を決定
    formElem.target = "";

    //ラジオボタン「感想」選択時の処理
    if(formElem.radio1[0].checked == true){
        var res = confirm("送信内容はご感想でよろしいですか？");
        if(res == true){
            formElem.action = "/cgi-bin/form_action.cgi?id=0"; //送信先を設定
            formElem.target = "blank";
        }else{
            return false;
        }
    }

    //ラジオボタン「質問」選択時の処理
    if(formElem.radio1[1].checked == true){
        var res = confirm("送信内容はご質問でよろしいですか？");
        if(res == true){
            formElem.action = "/cgi-bin/form_action.cgi?id=1"; //送信先を設定
        }else{
            return false;
        }
    }
}
```

HTML

```
<body>
  <form action="" id="form1" onsubmit="return sendForm()">
    <p>
    <input type="radio" name="radio1" checked="checked" />感想
    <input type="radio" name="radio1" />質問
    </p>
    <p><textarea name="text" cols="30" rows="5"></textarea></p>
    <p>
    <input type="submit" value="送信" />
    <input type="reset" value="クリア" />
    </p>
  </form>
</body>
```

Internet Explorer

ボタンのチェックによって送信先を変更します

参照
action プロパティ ・・・・・・・・・・・・・・・・・・・・ P.106
encoding プロパティ ・・・・・・・・・・・・・・・・・・ P.106
method プロパティ ・・・・・・・・・・・・・・・・・・・ P.106
target プロパティ ・・・・・・・・・・・・・・・・・・・・ P.106

FORM.SAMPLE-02

フォームの部品を参照する

フォームの空欄をチェックする関数です。[送信]ボタンがクリックされたとき(onsubmitイベント)に関数checkを呼び出し、フォーム内の全エレメントの入力を確認します。エレメントのタイプがtextであり、かつ未入力であるエレメントが存在する場合、警告ダイアログを表示してfalseを返します。

※onsubmitイベントで呼び出された関数でfalseが返された場合はフォームの内容は送信しません。

JavaScript

```
//フォーム内の未入力をチェックする関数
function check(){
  var formElem = document.getElementById("form1");
  for( i=0 ; i<formElem.length ; i++ ){ //フォーム内のエレメントの数だけ繰り返し
    //エレメントのタイプが"text"、かつ未入力の場合
    if((formElem.elements[i].type == "text") && (formElem.elements[i].
value == "")){
      alert("すべての欄に入力してください");
      return false; //falseを返す
    }
  }
  return true; //すべての項目を入力したらtrueを返す
}
```

HTML

```
<body>
  <form action="sample.cgi" id="form1" method="post" onsubmit="return
check()">
    <p>名前:<input type="text" name="name" size="28" /></p>
    <p>年齢:<input type="text" name="age" size="4" /></p>
    <p>住所:<input type="text" name="add" size="48" /></p>
    <p>電話:<input type="text" name="tel" size="18" /></p><hr />
    <p>
    <input type="submit" value="送信" />
    <input type="reset" value="クリア" />
    </p>
  </form>
</body>
```

Internet Explorer

[送信]ボタンをクリックしたときに入力欄に空欄があれば、警告ダイアログが表示され、送信が行われません

FORM.SAMPLE-03

選択されている項目を調べる

セレクトボックスの選択項目を調べ、選択項目によって処理を分けるサンプルです。「希望しない」選択時はメールアドレス入力用のテキストボックスと[確認]ボタンが無効状態になります。「希望する」を選択時はテキストボックスと[確認]ボタンが有効状態になります。

[確認]ボタンがクリックされると、テキストボックスの未入力チェック、メールアドレスであるかをチェックし、ダイアログを表示します。ダイアログ表示後、メールアドレスが正しくないもしくは未入力の場合はテキストボックスにフォーカスを移します。入力された文字列がある場合は選択状態となります。

JavaScript

```
var textElem;
//セレクトボックスの選択項目によってテキストボックスの有効・無効を切り替える関数
function changeSelect(){
    textElem = document.getElementById("text");
    var btnElem = document.getElementById("button");
    if(document.getElementById("select").selectedIndex == 0){
                                                    //選択項目をインデックスで判別
        //「希望しない」選択時の処理
        textElem.value = ""; //value値を空にする
        textElem.disabled = true; //テキストボックスを無効にする
        btnElem.disabled = true; //ボタンを無効にする
    }else{
        //「希望する」選択時の処理
        textElem.disabled = false; //テキストボックスを有効にする
        btnElem.disabled = false; //ボタンを有効にする
    }
}

//「確認」ボタンをクリックしたときの処理の関数
function clickButton(){
    //「希望する」を選択している場合のみ行う処理
    if(textElem.disabled == false){
        if(textElem.value == ""){ //未入力の場合
            alert("メールアドレスを入力して下さい。");
            textElem.focus(); //フォーカスを合わせる
        }else if(textElem.value.indexOf("@",0)<0){
                                        //入力された文字列に「@」がない場合
            alert("メールアドレスを正確に入力して下さい。");
```

```
            textElem.focus();
            textElem.select(); //入力された文字列を選択状態にする
        }else{
            alert("メールアドレスです。");
        }
    }
}
```

HTML

```
<body>
  <form action="" id="form1">
    <p>メールの送信を希望しますか？<br />
    <select id="select" onchange="changeSelect()">
      <option selected="selected">希望しない</option>
      <option>希望する</option>
    </select></p>
    <p>メールアドレス入力:<input type="text" id="text" disabled="disabled" />
    <input type="button" id="button" value="確認" onclick="clickButton()"
disabled="disabled" /> </p>
  </form>
</body>
```

Internet Explorer

❶セレクトボックスの選択項目で「希望しない」選択時はメールアドレス入力用のテキストボックス、[確認]ボタンが無効になります

❷メールアドレスに「@」が含まれていない場合、警告ダイアログが表示されます

参照

Elementsオブジェクト ・・・・・・・・・・・・・・・ P.108
lengthプロパティ ・・・・・・・・・・・・・・・・・・ P.108
typeプロパティ ・・・・・・・・・・・・・・・・・・・・ P.108

FORM.SAMPLE-04

フォームの内容を送信する

JavaScriptを使ってフォームの内容を送信するサンプルです。サンプル「フォームの部品を参照する」と内容はほぼ同じですが、このサンプルでは<input type ="submit" />で作成される送信ボタンの機能をsubmitメソッドで実現しています。

なお、フォームに入力されたデータを実際に送信するには、送信先や送信方法をHTML/XHTMLまたはJavaScriptを利用して別に設定し、また送信したデータを処理するためのCGIなどのプログラムが必要です。

JavaScript

```
//フォーム内の未入力をチェックする関数
function check(){
    var formElem = document.getElementById("form1");
    for( i=0 ; i<formElem.length ; i++ ){ //フォーム内のエレメントの数だけ繰り返し
        if((formElem.elements[i].type == "text") && (formElem.elements[i].
value == "")){ //エレメントのタイプが"text"、かつ未入力の場合
            alert("すべての欄に入力してください");
            return;
        }
    }
    formElem.submit(); //未入力がなければ送信する
}
```

HTML

```
<body>
  <form action="submit.html" id="form1" method="post">
    <p>
    名前：<input type="text" name="name" size="28" />
    年齢：<input type="text" name="age" size="4" />
    </p>
    <p>住所：<input type="text" name="add" size="48" /></p>
    <p>電話：<input type="text" name="tel" size="18" /></p>
    <hr/>
    <p>
    <input type="button" value="送信" onclick="check()" />
    <input type="button" value="クリア" onclick="reset()" />
    </p>
  </form>
</body>
```

Internet Explorer

[送信]ボタンをクリックしたときに入力欄に空欄があれば、警告ダイアログが表示され、送信が行われません

参照

submit メソッド …………………… P.110
reset メソッド …………………… P.110

EVENT.01

読み込み時や移動時に処理を行いたい

onload = ★ ページの読み込み時

onunload = ★ ページの切り替え時

★……実行する命令(関数や関数名)

形式 イベント

ページの内容が完全に読み込まれたときやページの切り替え時に処理を実行したい場合に使用します。

onloadイベント

ページや画像などデータの読み込みが完了したときに発生するイベントです。読み込み完了と同時に何らかの処理を行いたい場合に使用します。

onunloadイベント

他のページに移動するときに発生するイベントです。

文例

```
<body onload="timer1=setTimeout('msg', 3000)">
```

タイマーを設定し、その識別子をtimer1に代入します。ページの読み込みが完了したら、3秒おきに関数msgを呼び出します。

```
document.images[0].onload = msg;
```

1番目の画像の読み込みが完了したら、関数msgを呼び出します。

```
<body onunload="alert('またね！')">
```

他のページに移動するときに、「またね！」というダイアログを表示します。

Column　Eventオブジェクト

Eventはマウスやキーの状態などをあらわすオブジェクトです。イベントを取得したり、発生したイベントを参照できます。

Internet Explorerでは、EventオブジェクトをWindowオブジェクトのeventプロパティで扱うことができます。一方、FirefoxやNetscape、Operaでは、HTML/XHTMLのon～形式のイベントハンドラの内部でのみイベントオブジェクトが利用できるため、関数の引数としてEventオブジェクトを渡すのが一般的です。

たとえば以下のような形跡で指定します。

```
onclick="関数名(event)"
```

onclick=presskeyのようにメソッドを関数で置き換えるときは引数として渡されます。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照

画像の読み込みの完了を調べたい・・・・・・・・・・P.224
【SAMPLE】読み込み時に処理を行う・・・・・・・・P.148

EVENT.02

画像が読み込めないときに処理を行いたい

onabort = ★ 画像の読み込み中断時
onerror = ★ 画像の読み込み失敗時

★……実行する命令（関数や変数名）

形式 イベント

画像が読み込めない場合に発生するイベントです。読み込めない画像の有無を調べたり、その場合の処理を設定したりする際に使用します。

onabortイベント

画像の読み込みが中断されたときに発生するイベントです。必ず画像を表示させたい場合などに読み込みが中断されたタイミングで警告メッセージを表示できます。

onerrorイベント

画像ファイルが見つからないなどの理由で画像が表示できない場合に発生するイベントです。

文例

```
document.images[0].onabort =retry ;
```

1番目の画像の読み込みが中断された際、関数retryを呼び出します。

```
<img src="sample.jpg" alt="" onerror="alert('表示できません。')" />
```

画像の読み込みに失敗した際、「表示できません。」とダイアログを表示します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照
画像の読み込みの完了を調べたい ………… P.224
【SAMPLE】読み込み時に処理を行う ……… P.148

EVENT.03

サイズ変更時に処理を行いたい

onresize = ★

★……実行する命令(関数や[illegible]名)

形式 イベント

オブジェクトのサイズが変更されたときに発生するイベントです。サイズが変更されたときに処理を実行したい場合に使用します。ウィンドウやフレームで利用できます。

文例

```
<body onresize="alert('リサイズされました')">
```

ウィンドウのサイズが変更されたときに、ダイアログに「リサイズされました」と表示します。

▶ ブラ[illegible]対応表	IE10	IE9	[illegible]	Fx	[illegible]	Safari	[illegible]	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照
ウィンドウのサイズを変更したい・・・・・・・・・・P.080
【SAMPLE】フォーカスの移動時に処理を行う・・P.136

EVENT.04

フォーカスの移動時に処理を行いたい

onfocus = ★　フォーカスが合ったとき

onblur = ★　フォーカスが[illegible]れたとき

★……実行する命令(関数や[illegible]名)

形式　イベント

マウスカーソルや[Tab]キーによってフォーカスが[illegible]したときに発生するイベントです。ウィンドウやフレーム、フォームの[illegible]に設定できます。

onfocusイベント

フォーカスが合ったときに発生するイベントです。

onblurイベント

フォーカスが離れたときに発生するイベントです。

文例

```
<body onfocus="msg()">
```

ドキュメント内にフォーカスが合ったとき、関数msg()を実行します。

```
document.form1.elements[0].onblur = check;
```

form1の最初のエレメントからフォーカスが離れたとき、関数checkを呼び出します。

```
<input type="text" name="mail " size="40" onfocus="myFunc1()"
onblur="myFunc2()">
```

入力フィールドがフォーカスされたら関数myFunc1()、入力フィールドからフォーカスが離れたら関数myFunc2()を呼び出します。

▶ ブラウザ対応表	IE10	IE9	IE8	[illegible]	[illegible]	Safari	Opera	[illegible]	Android
	○	○	○	○	○	○	○	○	○

参照
自動的にフォーカスを移動させたい・・・・・・・・P.115
【SAMPLE】フォーカスの移動時に処理を行う・・P.136

EVENT.05

マウスオーバー時に処理を行いたい

onmouseover = ★ マウスオーバー時
onmouseout = ★ マウスカーソルが離れたとき

★……実行する命令（関数や関数名）

形式 イベント

マウスカーソルがオブジェクト上に重なったとき、またはオブジェクトから離れたときに発生するイベントです。ページ上のほとんどの要素に利用することができます。

onmouseoverイベント

オブジェクト上にマウスカーソルが重なったときに発生するイベントです。

onmouseoutイベント

オブジェクトからマウスカーソルが外れたときに発生するイベントです。

文例

```
<a href="javascript:void(0)" onmouseover="mover()">
```

リンク上にマウスカーソルが重なったら、関数mover()を呼び出します。

```
document.links[0].onmouseout = msg;
```

最初のリンクからマウスカーソルが離れたら、関数msgを呼び出します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照

マウスクリック時に処理を行いたい……… P.134
キー操作によって処理を行いたい………… P.139
【SAMPLE】マウス操作時に処理を行う…… P.138

マウスクリック時に処理を行いたい

onclick = ★	マウスクリック時
ondblclick = ★	マウスダブルクリック時
onmousedown = ★	マウスダウン時
onmouseup = ★	マウスアップ時

★……実行する命令(関数)

形式 イベント

マウスのボタンがクリックされたときに発生するイベントです。ページ上のほとんどの要素で利用できます。

onclickイベント

マウスがクリックされたときに発生するイベントです。発生のタイミングはonmouseupイベントと同じです。

ondblclickイベント

マウスのボタンがダブルクリックされたときに発生するイベントです。

onmousedownイベント

マウスのボタンがクリックされたときに発生するイベントです。

onmouseupイベント

マウスのボタンが離されたときに発生するイベントです。

文例

```
<input type="button" value="OK" onclick="myFunc()" />
```

ボタンがクリックされたら関数myFunc()を呼び出します。

```
<a href="javascript:void(0)" onmousedown="alert('はずれ')">
```

リンク部分でマウスのボタンがクリックされたら、ダイアログに「はずれ」と表示します。

```
document.myImg1.onmouseup = myFunc;
```

画像名myImg1上でマウスのボタンが離れたら、関数myFuncを呼び出します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	×	×

マウスオーバー時に処理を行いたい ········ P.133
キー操作によって処理を行いたい ·········· P.139
【SAMPLE】マウス操作時に処理を行う ······ P.138

EVENT.07

コンテキストメニューを表示させないようにしたい

oncontextmenu = ★　　コンテキストメニュー表示時

★……実行する命令(関数や[illegible]名)

形式　イベント

マウスの右ボタンがクリックされたとき(コンテキストメニューが表示される前)に発生するイベントです。このイベントからの戻り値をfalseにすると、マウスを右クリックした際に表示されるコンテキストメニューを表示しません。

文例

```
<body oncontextmenu="return false">
```
コンテキストメニューを表示しません。

ブラウザ対応表	IE10	[illegible]	[illegible]	Fx	Chrome	Safari	[illegible]	iOS6	Android
	○	○	○	○	○	○	○	×	×

参照　【SAMPLE】マウス操作時に処理を行う・・・・・・P.138

EVENT.08

フォームの送信／リセット時に処理を行いたい

onsubmit = ★　　フォーム送信時
onreset = ★　　フォームリセット時

★……実行する命令(関数や関数名)

式 イベント

フォームが送信またはリセットされたときに発生するイベントです。実行する命令の戻り値がfalseの場合は送信やリセットを行いません。

onsubmitイベント

submitボタンがクリックされたときに発生するイベントです。

onresetイベント

resetボタンがクリックされたときに発生するイベントです。

文例

```
document.form1.onsubmit = formCheck;
```

フォームが送信されたら関数formCheckを呼び出します。

```
<form onreset="return confirm('リセットしますか？')">
```

送信ボタンが押されたときに確認ダイアログを表示します。[OK]ボタンが押されたら送信を行い、[キャンセル]ボタンが押された場合は送信を行いません。

Column　条件によってフォームの送信を中止する

フォームの送信をJavaScriptで制御するメリットの一つは、未記入の項目があるとき、フォームを送信しないようにできることです。送信しないようにするには、onsubmitイベントの処理としてreturnステートメントでfalseを返すようにします。

入力の確認などをしてからフォームを送信させた場合はreturnの後に関数名を指定して、その関数の中で入力が誤っていたらfalseを返すようにしてください。たとえば

```
function send_mail() {
    return false;
}
```

というfalseを返す関数を作成して、<form>タグのonsubmitイベントを

```
onsubmit="return send_mail()"
```

のように記述すると、フォームを送信しません(常にfalseを返します)。

この応用として以下のように記述すれば、text1に文字列が入力されている場合だけ送信できるようになります(document.form1はthisと書き換えることができます)。

```
<form name="form1" action="/cgi-bin/test.cgi" onsubmit="return document.
form1.text1.value != "">
    <input type="text" name="text1" />
    <input type="submit" value="送信" />
</form>
```

ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

フォームの内容をリセット／送信したい・・・・・P.110
【SAMPLE】フォーム操作時に処理を行う・・・・P.130

EVENT.09

フォーム操作時に処理を行いたい

onchange = ★ 変更時
onselect = ★ 入力フィールド選択時

★……実行する命令(関数や[illegible]名)

式 イベント

セレクトメニュー(select要素)やテキスト入力フィールド(textarea要素、input要素の属性がtype="text"の場合)などのフォームの部品の状態や内容が変化したときに発生するイベントです。

onchangeイベント

フォームの部品の状態や内容が変更されたときに発生するイベントです。select、textarea、input要素(type="text"の場合)などに設定することで、セレクトメニューの選択項目やテキスト入力フィールドの文字列が変更されたときに発生させることができます。

onselectイベント

選択時に発生するイベントです。textarea要素、input type="text"に設定することで、テキスト入力欄の文字列が選択されたときに発生させることができます。

文例

```
<select onchange="idou()">
```

セレクトメニューの項目が変更されたら、関数idou()を呼び出します。

```
document.form1.text1.onselect = myFunc;
```

text1の文字列が選択されたら、関数myFuncを呼び出します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

どの項目が選択されているかを調べたい ····· P.112
【SAMPLE】フォーム操作時に処理を行う ···· P.130

EVENT.10

キー操作によって処理を行いたい

```
onkeydown = ★
onkeypress = ★
onkeyup = ★
```

キーが押されたとき
キーが押されているとき
キーが離されたとき

★……実行する命令(関数や関数名)

形式 イベント

キーの操作状態によって発生するイベントです。同じオブジェクトに設定している場合は、onkeydown→onkeypress→onkeyupイベントの順に発生します。ページ上のほとんどの要素で利用可能です。なお、同じ操作でもブラウザの種類によって、発生するイベントの種類や発生順序が異なる場合があります。

onkeydownイベント

キーが押されたときに発生するイベントです。

onkeypressイベント

キーが押されている間、[illegible]続的に発生するイベントです。

onkeyupイベント

キーが押され、その後、離されたときに発生するイベントです。

文例

```
<body onkeydown="alert('呼んだ？')">
```

キーが押されたら、「呼んだ？」というダイアログを表示します。

```
document.onkeypress = kpress;
```

キーが押されている間、関数kpressを呼び出します。

```
<a href="#" onkeyup="kup()">
```

キーが離されたら、関数kup()を呼び出します。

▶ ブラウザ対応表	IE10	[illegible]	IE8	Fx	Chrome	Safari	[illegible]	iOS6	Android
	○	○	○	○	○	○	○	○	○

マウスオーバー時に処理を行いたい ········ P.133
マウスクリック時に処理を行いたい ········ P.134
【SAMPLE】押されたキーのキーコードを取得する ·· P.156

EVENT.11

押されたキーの キーコードを取得したい

★.keyCode

キーコードを参照

★……Eventオブジェクト

形式 プロパティ

入力されたキーのキーコード（文字コード）を参照します。キーコードから文字列を取得するには、StringオブジェクトのfromCharCodeメソッド（p.222）を使用してください。

文例

```
alert(event.keyCode);
```

押されたキーのキーコードをダイアログに表示します。

▶ ブラウザ対応表	IE10	IE9	IE8	[illegible]	Chrome	Safari	Opera	[illegible]	Android
	○	○	○	○	○	○	○	○	○

イベントの情報を取得したい…………P.141
文字コードを扱いたい…………P.203
【SAMPLE】押されたキーのキーコードを取得する……P.156

EVENT.12

イベントの情報を取得したい

★.target
★.type

★……Eventオブジェクト

形式 プロパティ

targetプロパティ

画像やフォームエレメントなど、イベントの発生元となるオブジェクトを返します。

typeプロパティ

発生したイベントの種類を参照します。イベントの種類はイベントハンドラ名から先頭のonを削除した部分の文字列になります(click、mousedownなど)。

文例

```
myEvent = event.type;
```

イベントの種類を変数myEventに代入します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
target	×	×	×	○	○	○	○	○	○
type	○	○	○	○	○	○	○	○	○

参照

押されたキーのキーコードを取得したい・・・・・P.140
イベントが発生した位置を調べたい・・・・・・・・・P.128
【SAMPLE】イベントの情報を取得する・・・・・・P.158

EVENT.13

イベントが発生した位置を調べたい

★.x	マウスのx座標を参照
★.y	マウスのy座標を参照
★.clientX	マウスの表示領域上のx座標を参照／設定
★.clientY	マウスの表示領域上のy座標を参照／設定
★.pageX	マウスのページ上のx座標を参照
★.pageY	マウスのページ上のy座標を参照
★.screenX	マウスの画面上のx座標を参照
★.screenY	マウスの画面上のy座標を参照

★……Eventオブジェクト

形式　プロパティ

ページ上および画面上における、イベント発生時のマウスの位置を調べます。clientXとclientYプロパティではブラウザの表示領域の左上隅からの相対座標を参照／設定するのに対し、pageXとpageYプロパティでは作成されるページ全体の左上隅からの相対座標を参照します。

文例

```
alert(event.x + "," + event.y);
```
イベント発生時におけるマウスのx座標とy座標をダイアログに表示します。

```
alert(event.clientX);
```
イベント発生時における、表示領域上のマウスのx座標をダイアログに表示します。

```
cY = event.clientY;
```
イベント発生時における、表示領域上のマウスのy座標を変数cYに代入します。

```
document.eForm.px.value = event.pageX;
```
エレメント名pxの値をイベント発生時におけるマウスのページ上のx座標とします。

```
document.eForm.py.value = event.pageY;
```
エレメント名pyの値をイベント発生時におけるマウスのページ上のy座標とします。

```
sX = event.screenX;
```
イベント発生時におけるマウスの画面上のx座標を変数sXに代入します。

```
if(event.screenY >= 250)myFunc();
```
イベント発生時におけるマウスの画面上のy座標が250ピクセル以上の場合、関数myFunc()を呼び出します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
x、y	○	○	○	×	×	○	○	○	×
pageX、pageY	×	×	×	○	○	○	○	○	○
その他	○	○	○	○	○	○	○	○	○

イベントの情報を取得したい・・・・・・・・・・・・・P.141
[SAMPLE] イベントが発生した位置を調べる・・P.160
[SAMPLE] マウスの動きに合わせて画像を動かす・・P.162

EVENT.SAMPLE-01

フォーム操作時に処理を行う

フォームで選択された内容によって自動的に金額を計算し、送信するサンプルです。フォーム上で「もも」「ぶどう」それぞれの個数が変更される(onchangeイベント)とcalcPrice関数を呼び出し、合計金額を自動計算して表示します。

[送信]ボタンがクリックされる(onsubmitイベント)と確認ダイアログに合計金額を表示し、[OK]ボタンで送信し、[クリア]ボタンで送信をキャンセルします。

[クリア]ボタンがクリックされる(onresetイベント)とフォーム内容のリセットに関する確認ダイアログを表示し、[OK]ボタンでフォーム内容をリセット、[キャンセル]ボタンでフォーム内容を保持したままフォーム画面に戻ります。

JavaScript

```
//合計金額を計算する関数
function calcPrice(){
    var momo = document.getElementById("momo").selectedIndex * 4000;
    var budou = document.getElementById("budou").selectedIndex * 3000;
    document.getElementById("output").value = momo + budou;
}

//送信ボタンをクリックしたときの処理をする関数
function submitForm(){
    if(document.getElementById("output").value == "0"){
        alert("金額は0円です。");
        return false;
    }
    var res = confirm("金額は" + document.getElementById("output").value + "
円です。¥nよろしいですか？");
    if(!res){
        return false; //SUBMIT処理をキャンセルします
    }
    return true;
}

//リセットボタンをクリックしたとき処理をする関数
function resetForm(){
    var res = confirm("フォーム内容をリセットします。¥nよろしいですか？");
    if(!res){
        return false; //リセットの処理をキャンセルします
    }
```

```
    return true;
}
```

HTML

※ボーダーや内容領域のサイズは外部CSSで指定しています

```
<body>
    <form action="submit.html" onsubmit="return submitForm()"
onreset="return resetForm()">
        <table>
            <tr><th>商品</th><th>単価</th><th>注文数</th></tr>
            <tr><td>もも</td><td>4000円</td>
            <td><select id="momo" onchange="calcPrice()">
                <option>0</option>
                <option>1</option>
                <option>2</option>
                <option>3</option>
            </select>箱</td></tr>
            <tr><td>ぶどう</td><td>3000円</td>
            <td><select id="budou" onchange="calcPrice()">
                <option>0</option>
                <option>1</option>
                <option>2</option>
                <option>3</option>
            </select>箱</td></tr>
            <tr><td colspan="2">合計金額</td>
            <td><input type="text" id="output" value="0" size="6" />円</td></
tr>
        </table>
        <p>
        <input type="submit" id="submit" value="送信" />
        <input type="reset" id="reset" value="クリア" />
        </p>
    </form>
</body>
```

Internet Explorer

❶注文数によって自動計算し、合計金額を算出します

❷[送信]ボタンをクリックすると、確認ダイアログに合計金額が表示されます

❸[クリア]ボタンをクリックすると、確認ダイアログが表示されます

参照

onsubmit イベント …… P.136
onreset イベント …… P.136
onchange イベント …… P.138

EVENT.SAMPLE-02

読み込み時に処理を行う

ページの読み込み終了時(onloadイベント)、他のページへの移動時(onunloadイベント)、画像読み込み失敗時(onerrorイベント)にメッセージを表示します。なお、imgタグのsrc属性に存在しないファイル(dog.gif)が指定されているため、画像の読み込みが失敗します(onerrorイベント発生)。onerrorイベント発生時にimgError関数によってメッセージを表示し、src属性に存在するファイル(cat.gif)を指定し直しています。

JavaScript

```
//イベントの情報を参照する関数
function imgError(){
    res = confirm("画像の読み込みエラーです。¥n再度ページを読み込みます");
    if(res == true){ //confirmダイアログで「OK」をクリック
        document.getElementById("img").src = "images/cat.gif";
                                                  //imgタグのsrc属性を変更
    }
}
```

HTML

```
<body onload="alert('読み込み完了しました。')" onunload="alert('ページを移動します。')" >
    <p><img id="img" src="images/dog.gif" alt=""  onerror="imgError()"/></p>
</body>
```

Internet Explorer

ページを読み込むとき（画像の読み取りに失敗し、onerrorイベントが発生）

ページの読み込みが完了したとき（onloadイベント発生時）

別のページへ移動するとき（onunloadイベント発生時）

EVENT.SAMPLE-03

フォーカスの移動時に処理を行う

フォームの入力をチェックするサンプルです。[e-mail]入力欄からフォーカスが離れたとき（onblurイベント）にcheckAdr関数を呼び出し、入力内容（入力内容に「@」が含まれるか）を確認します。正しく入力されていなかった場合、警告ダイアログを表示し、テキスト入力フィールドに「@」を追加してフォーカスを[e-mail]入力欄に戻します。入力文字列があれば選択状態にします。

また、ページロード時（window.onload）にchangeSize関数を呼び出し、ウィンドウサイズと表示位置を指定しています。

ウィンドウサイズ変更時（window.onresize）には、resize関数によってページロード時と同じウィンドウサイズと表示位置に戻すように指定しています。ただし、Internet ExplorerとOperaでは正常に動作しないので、FirefoxとChromeのみを判別して処理をしています。

JavaScript

```
window.onload = changeSize;
window.onresize = resize;

//ウィンドウサイズを設定する関数
function changeSize(){
    resizeTo(screen.availWidth / 2, screen.availHeight / 2);
                                              //ウィンドウサイズの設定
    moveTo(screen.availWidth / 4, screen.availHeight / 4);
                                              //ウィンドウ表示位置の設定
}

//フォーカスが離れたときの処理の関数
function checkAdr(check){
    var textElem = document.getElementById("email");
    if(textElem.value.indexOf("@",0) < 0){ //入力された文字列に「@」がない場合
        alert("メールアドレスを正確に入力して下さい。");
        textElem.value += "@";
        textElem.focus();
        textElem.select(); //入力された文字列を選択状態にする
    }
}

//サイズ変更時に呼ばれる関数
function resize(){
    //Firefox,Chromeの場合のみ
```

```
    if(navigator.appName.indexOf("Netscape") > -1){
        resizeTo(screen.availWidth / 2, screen.availHeight /2 );
                                                    //ウィンドウサイズの設定
        moveTo(screen.availWidth / 4, screen.availHeight / 4);
                                                    //ウィンドウ表示位置の設定
    }
}
```

HTML

```
<body>
    <form action="" id="form1" >
        <p>お名前<input type="text" id="name" size="30" /></p>
        <p>e-mail:<input type="text" id="email" size="30"
onblur="checkAdr()"/></p>
    </form>
</body>
```

Internet Explorer

❶入力したメールアドレスに「@」が入っていない場合に[e-mail]入力欄からフォーカスを離すと、警告ダイアログが表示されます

❷[e-mail]入力欄に「@」が挿入されます

参照
onresize イベント ･････････････････････ P.131
onblur イベント ･･････････････････････ P.132

EVENT.SAMPLE-04

マウス操作時に処理を行う

マウス操作時に処理を行うサンプルです。画像の下にある[down & up]ボタンがダウン(mousedownイベント)、アップ(mouseupイベント)されるたびに画像を切り替えます。マウスカーソルを画像に合わせたとき(mouseoverイベント)、外したとき(mouseoutイベント)も画像を切り替えます。また、画像をダブルクリックする(dblclickイベント)と、ダイアログを表示し、簡単なアニメーションのような動きを実現します。

JavaScript

```
//ページロード時にimgの配列を作成する関数
function pageLoad(){
    var imgNum = 4; //イメージの数
    images = new Array(); //文字列(imgタグのsrc属性に設定用)配列を作成
    imgElem = document.getElementById("img");
    //イメージの数だけ
    for(i = 0 ; i < imgNum ; i++){
        images[i] = "images/anime" + i + ".gif"; //配列の要素に順に代入
    }
}

//マウスオーバー時の処理の関数
function mouseOver(){
    imgElem.src = images[3]; //imgタグのsrc属性を変更
}

//マウスアウト時の処理の関数
function mouseOut(){
    imgElem.src = images[0]; //imgタグのsrc属性を変更
}

//マウスダウン時の処理の関数
function mouseDown(){
    imgElem.src = images[1]; //imgタグのsrc属性を変更
}

//マウスアップ時の処理の関数
function mouseUp(){
    imgElem.src = images[0]; //imgタグのsrc属性を変更
}
```

```
//マウスダブルクリック時の処理の関数
function dblClick(){
    count = 0; //アニメーション用のカウント変数
    alert("アニメーションを開始します。");
    timer1 = setInterval('animation()', 500); //タイマーをセット
}

//画像をアニメーションさせる処理の関数
function animation(){
    imgElem.src = images[count]; //0.5秒ごとに画像を切り替えます
    if(count >= 3){ //4枚目まで画像が切り替わったら
        clearInterval(timer1); //タイマーを削除します
    }
    count++; //カウント変数を+1します
}

//マウス右クリック時の処理の関数
function onRClick(){
    alert("右クリックは無効です。");
    return false; //falseを返すと右クリックが無効になります
}
```

HTML

```
<body id="body" onload="pageLoad()">
    <p><img id="img" src="images/anime0.gif" alt="" ondblclick="dblClick()"
onmouseover="mouseOver()" onmouseout="mouseOut()" oncontextmenu="return
onRClick()" /></p>
    <form action="">
        <p><input type="button" value="down & up" onmousedown="mouseDown()"
onmouseup="mouseUp()" /></p>
    </form>
</body>
```

Internet Explorer

❶マウスカーソルが画像に合わさったときに画像が切り替わります

❷マウスを操作していない状態

❸[down & up]ボタンをクリックすると、画像が切り替わります

❹画像をダブルクリックすると、ダイアログが表示されます。さらに[OK]ボタンをクリックすると、次々に画像を切り替わり、アニメーションのような動きを実現します

❺右クリックすると、ダイアログが表示されます

参照

onmouseover イベント ………………… P.133
onmouseout イベント ………………… P.133
ondblclick イベント ………………… P.134
onmousedown イベント ………………… P.134
onmouseup イベント ………………… P.134
oncontextmenu イベント ………………… P.135

EVENT.SAMPLE-05

押されたキーのキーコードを取得する

SAMPLE SCRIPT

キーボードの入力値に関する情報を表示するサンプルです。テキスト入力フィールドでキーボードが押された（onkeydownイベント）ときに、pressKey関数を呼び出し、押されたキーのキーコードを参照し、テキストエリアに表示します。また、そのキーコードをfromCharCodeメソッドで文字に変換したものも同時に表示します。

なお、ページがロードされたときにonkeydownイベントの設定の他に、テキスト入力フィールドのime-mode属性をdisabledに設定することで日本語入力を不可にしています(Internet Explorerのみ)。

JavaScript

```
var keyNum;
var keyChar;
var keyStr ="";

//入力モードを制限する関数
function loadPage(){
    document.getElementById("text").style.imeMode = "disabled";
                                    //テキストボックスの日本語入力をオフにする
    document.getElementById("text").onkeydown = pressKey;
}

//押したキーのキーコードを出力する関数
function pressKey(e){
    if(e){
        keyNum = e.keyCode; //キーコード
    }else{
        keyNum = event.keyCode; //キーコード
    }
    keyChar = String.fromCharCode(keyNum); //キーコードを文字に変換
    keyStr = keyStr + "キー：" + keyChar + " / " + "キーコード：" + keyNum + "¥n";
    document.getElementById("log").value = keyStr; //テキストエリアに出力
}

//入力されている値をクリアする関数
function clickClear(){
    keyStr = "";
    document.getElementById("text").value = "";
    document.getElementById("log").value = "";
}
```

HTML

```
<body onload="loadPage()">
  <form action="" id="form1">
    <p>
    入力して下さい。<br />
    <input type="text" id="text" /><br />
    <textarea id="log" rows="6" cols="50"></textarea><br />
    <input type="button" value="クリア" onclick="clickClear()" />
    </p>
  </form>
</body>
```

Internet Explorer

[A][B][C][Shift]キーを順番に押した場合

参照
onkeydown イベント ・・・・・・・・・・・・・・・・・ P.139
keycode プロパティ ・・・・・・・・・・・・・・・・・ P.140

EVENT.SAMPLE-06

イベントの情報を取得する

イベントの発生したオブジェクト、イベントの種類をダイアログに表示するサンプルです。ページ上のリンクおよびフォームのボタンがクリックされたとき(onclickイベント)、関数getEventを呼び出します。その際、引数としてeventを渡します。関数getEventでは、そのイベントの発生元のオブジェクトとイベントの種類を警告ダイアログに表示します。なお、InternetExplorerはtargetプロパティに対応していないため、undefinedを返すので、その処理をif構文で分岐させています。

JavaScript

```
//イベントの情報を参照する関数
function getEvent(e){

    if(e.target == undefined){ //targetプロパティ非対応の場合

        alert("発生元オブジェクトは参照できません" + "¥nイベントの種類は" + e.type);
        return;

    }

    alert("発生元オブジェクトは" + e.target + "¥nイベントの種類は" + e.type);

}
```

HTML

```
<body>
    <p><a href="http://www.ank.co.jp/" onclick="getEvent(event)">株式会社アンク</a></p>
    <form action="">
        <p><input type="button" value="イベント情報"
onclick="getEvent(event)"/></p>
    </form>
</body>
```

Internet Explorer8

Internet Explorer8はtargetプロパティに対応していないため、undefinedが返されます

Firefox

発生元オブジェクトは[object HTMLInputElement]
イベントの種類はclick

ボタンのクリック時

発生元オブジェクトはhttp://www.ank.co.jp/
イベントの種類はclick

リンクのクリック時

type イベント ······················ P.141
target イベント ···················· P.141

EVENT.SAMPLE-07

イベントが発生した位置を調べる

マウスがクリックされた位置を座標で表示するサンプルです。マウスのボタンがクリックされたとき(onmousedownイベント)、mDown関数を呼び出し、そのときのマウスの位置(座標)をフォームに表示しています。各ブラウザで対応していないプロパティは値がundefinedとなります。

JavaScript

```
document.onmousedown = mDown; //onmousedownイベントにmDown関数を登録

//座標を取得する関数
function mDown(e){
    //変数の宣言
    var sxElem = document.getElementById("sx");
    var syElem = document.getElementById("sy");
    var pxElem = document.getElementById("px");
    var pyElem = document.getElementById("py");
    var xElem = document.getElementById("x");
    var yElem = document.getElementById("y");
    //Internet Explorer以外の処理
    if(e){
        sxElem.value = e.screenX;
        syElem.value = e.screenY;
        pxElem.value = e.pageX;
        pyElem.value = e.pageY;
        xElem.value = e.x;
        yElem.value = e.y;
    //Internet Explorerの処理
    }else{
        sxElem.value = event.screenX;
        syElem.value = event.screenY;
        pxElem.value = event.pageX;
        pyElem.value = event.pageY;
        xElem.value = event.x;
        yElem.value = event.y;
    }
}
```

HTML

※レイアウトは外部CSSで指定しています

```
<body>
  <form action="">
    <table>
      <tr>
        <td>モニタ上のX座標(screenX)</td>
        <td><input type="text" id="sx" /></td>
        <td>モニタ上のY座標(screenY)</td>
        <td><input type="text" id="sy" /></td>
      </tr>
      <tr>
        <td>ページ上のX座標(pageX)</td>
        <td><input type="text" id="px" /></td>
        <td>ページ上のY座標(pageY)</td>
        <td><input type="text" id="py" /></td>
      </tr>
      <tr>
        <td>前面のX座標(x)</td><td><input type="text" id="x" /></td>
        <td>前面のY座標(y)</td><td><input type="text" id="y" /></td>
      </tr>
    </table>
  </form>
</body>
```

Internet Explorer

クリックした位置の座標が表示されます。ブラウザに対応していないプロパティではundefinedが返されます

参照

x プロパティ …… P.142
y プロパティ …… P.142
pageX プロパティ …… P.142
pageY プロパティ …… P.142
screenX プロパティ …… P.142
screenY プロパティ …… P.142

EVENT.SAMPLE-08

マウスの動きに合わせて画像を動かす

マウスの動きに合わせて画像が動くサンプルです。

まずCSSファイルではposition:absolute:を指定します。これはマウスカーソルの座標に対し、画像を表示する領域の絶対値を確保するものです。マウスが動かされると、onmousemoveイベントに設定しているmoveMouse関数を呼び出し、イベントが発生した座標(マウスの位置)を取得します。moveMouse関数ではイベント引数を取得できるかどうかでブラウザを判別し、処理を振り分けています。imgMove関数では、外部CSSファイルのleft, topプロパティを書き換えており、これによって画像の位置が指定されます。このimgMove関数はタイマーで150ミリ秒ごとに呼び出されているので、マウスの動きに合わせて画像が移動するような動きが実現できます。なお、タイマーの設定は最初の1回のみ必要なので、初回のみの動作になるように変数flagを設定しています。

JavaScript

```
//変数の宣言
var imgElem;
var x;
var y;
var flag = true;

//ページロード時の処理
function loadPage(){
   imgElem = document.getElementById("img");
   document.onmousemove = moveMouse;
}

//マウスが動いたときの処理
function moveMouse(e){
   if(e){ //引数が取得できるとき(Firefoxなど)
      x = e.pageX;
      y = e.pageY;
   }else{ //引数が取得できないとき(IEなど)
      x = event.clientX;
      y = event.clientY;
   }
   if(flag){ //最初のみ実行する処理
      flag = false;
      setInterval("imgMove()",150); //タイマーをセット
   }
```

```
}

//画像の位置を設定する処理
function imgMove(){
    imgElem.style.left = x + document.body.scrollTop  +"px";
    imgElem.style.top = y + document.body.scrollLeft  + "px";
}
```

HTML

※レイアウトは外部CSSで指定しています

```
<body onload="loadPage()">
    <div id="img">
        <img src="images/ico_clover.gif" alt="" />
    </div>
</body>
```

CSS

```
div {
    position: absolute;
    top: -50px;
    left: -50px;
    z-index: 1;
}
```

マウスの動きに合わせて画像が動きます

参照

clientX プロパティ ・・・・・・・・・・ P.142
clientY プロパティ ・・・・・・・・・・ P.142
pageX プロパティ ・・・・・・・・・・ P.142
pageY プロパティ ・・・・・・・・・・ P.142

一定時間後に処理を行いたい

● = ★.setTimeout(◆, ▲) タイマーを設定
★.clearTimeout(●) タイマーを解除

●……タイマー識別子を格納する変数【省略可】※clearTimeoutを併用する場合は省略不可
★……Windowオブジェクト(ウィンドウ名)【省略可】
◆……実行する命令(関数)
▲……指定時間(ミリ秒)

 メソッド

指定した時間後に命令を実行します。

setTimeoutメソッド

タイマーの設定を行うメソッドです。setTimeoutメソッドで設定したタイマーは、指定時間が経過すると1回だけ関数を呼び出します。時間はミリ秒(1／1000秒)単位で指定し、たとえば1000を指定すると1秒後に動作します。

clearTimeoutメソッド

clearTimeoutメソッドは、setTimeoutメソッドで設定したタイマーを解除します。設定時にタイマー識別子を変数●で取得し、その変数を使ってタイマーを解除します。

文例

```
timer1 = setTimeout("msg()", 3000);
```
関数msg() を3秒後に呼び出します。

```
clearTimeout(timer1);
```
setTimeout メソッドで設定したタイマーを解除します。

ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

一定時間ごとに処理を行いたい ············ P.165

一定時間ごとに処理を行いたい

● = ★.setInterval(◆, ▲)　タイマーを設定
★.clearInterval(●)　タイマーを[illegible]除

●……タイマーの[illegible]を格納する[illegible]【省略可】※clearIntervalを併用する場合は省略不可
★……Windowオブジェクト(ウィンドウ名)【省略可】
◆……実行する命令(関数)
▲……指定時間(ミリ秒)

[illegible]式　メソッド

指定した時間ごとに繰り返し命令を実行させるには、setIntervalメソッドを利用します。

setIntervalメソッド

タイマーの設定を行うメソッドです。setIntervalメソッドで設定したタイマーは、[illegible]定した時間ごとに関数を呼び出し、clearIntervalメソッドで解除されない限り停止しません。時間はミリ秒(1／1000秒)単位で指定し、たとえば1000を指定すると1秒ごとに動作します。

clearIntervalメソッド

setIntervalメソッドで設定したタイマーの解除を行います。設定時にタイマー識別子を変数●で取得し、その変数を[illegible]ってタイマーを解除します。

文例

```
timer2 = setInterval("msg()", 500);
```
関数msg()を0.5秒ごとに呼び出します。

```
clearInterval(timer2);
```
setIntervalメソッドで設定したタイマーを解除します。

▶ブラウザ対応表	IE10	IE9	[illegible]	Fx	[illegible]	Safari	[illegible]	[illegible]	Android
	○	○	○	○	○	○	○	○	○

参照
一定時[illegible]に処理を行いたい・・・・・・・・・・・・・・P.164
【SAMPLE】一定時間ごとに処理を行う・・・・・・P.166

TIMER.SAMPLE-01

一定時間ごとに処理を行う

ページが読み込まれると0.01秒ごとにcountTime関数を呼び出し、経過時間を表示します。[停止]ボタンがクリックされると、タイマーを解除して(経過時間の表示が停止します)、ボタンの表示を[再開]に切り替えます。その状態でボタンがクリックされると、今度はタイマーをセットし、カウントを再開します。このときボタンの表示は[停止]に戻ります。

JavaScript

```
var hour = 0;
var min = 0;
var sec = 0;
var mSec = 0;
var strH;
var strM;
var strS;
var strMS;
function countTime(){ //経過時間を表示する関数
    mSec++; //mSecに1を加える
    if(mSec == 100){ //mSecが100になったら
        sec++; //secに1を加える
        mSec = 0; //mSecを0に戻す
        if(sec == 60){ //secが60になったら
            min++; //minに1を加える
            sec = 0; //secを0に戻す
            if(min == 60){ //minが60になったら
                hour++; //hourに1を加える
                min = 0; //minを0に戻す
            }
        }
    }
    //値を表示用変数に代入
    strH = hour;
    strM = min;
    strS = sec;
    strMS = mSec;
    //1桁の場合は頭に"0"をつける
    if(hour < 10)strH = "0" + strH;
    if(min < 10)strM = "0" + strM;
    if(sec < 10)strS = "0" + strS;
```

```
    if(mSec < 10)strMS = "0" + strMS;
    document.getElementById("time").value = strH + ":" + strM + ":" + strS +
":" + strMS;
}
function clickButton(){ //ボタンがクリックされたときの処理
    var btnElem = document.getElementById("button");
    if(btnElem.value == "停止"){ //「停止」ボタンクリック時
        clearInterval(timer1); //タイマーをクリア
        btnElem.value = "再開"; //ボタンの表示を「再開」に変更
    }else{ //「再開」ボタンクリック時
        timer1 = setInterval("countTime()", 10); //タイマーを再セット
        btnElem.value = "停止"; //ボタンの表示を「停止」に変更
    }
}
```

HTML

```
<body onload="timer1=setInterval('countTime()',10)" onunload="clearInterval
(timer1)">
<form action="" >
<p>経過時間：<input type="text" id="time" value="00:00:00:00" /><br />
<input type="button" id="button" value="停止" onclick="clickButton()" /></p>
</form>
</body>
```

Internet Explorer

❶ページが読み込まれると、経過時間のカウントが始まります

❷[停止]ボタンをクリックすると、タイマーが解除され、ボタンの表示が[再開]に切り替わります

参照
setInterval メソッド ・・・・・・・・・・・・・・・・・・ P.164
clearInterval メソッド ・・・・・・・・・・・・・・・・ P.164

ARRAY.01

配列を使いたい

★ = new Array(◆)	新しい配列を作成
★ = new Array(▲, ▲, .., ▲)	配列を作成しデータを格納
★.length	配列の要素数を参照／設定

★……Arrayオブジェクト
◆……配列の要素数［省略可］
▲……配列のデータ［省略可］

形式 オブジェクト(Array)、プロパティ(length)

配列は複数の変数を1つにまとめたもので、Arrayはこの配列を扱うためのオブジェクトです。配列を構成する変数を要素といいます。HTMLの要素とは意味が異なるので、注意してください。

Arrayオブジェクト

新しい配列を作成する際はnewステートメント(p.039)を使用します。配列の要素数をあらかじめ指定するには次のようになります。

```
new Array(◆)
```

要素のデータを指定するには次のようになります。

```
new Array(▲, ▲, .., ▲)
```

要素数や要素のデータを指定せずにnew Array()という形式で作成した配列は、後から加えた要素の数に合わせて自動的に配列の大きさを変更します。　配列の個々の要素を参照するには、"配列名[参照番号]"のように記述します。参照番号の部分は添字と呼ばれ、0からの連番になります。つまり、myArrayという配列の3番目の要素を参照したい場合は次のようになります。

```
myArray[2]
```

添字には参照番号の代わりに文字列を使うこともでき、このような配列を連想配列といいます。添字が英文字列の場合は、オブジェクト名と添字を「.」(ピリオド)でつないで表記しても

同じ意味になります。

lengthプロパティ

配列★の要素数を参照／設定します。

文例

```
myArray = new Array();
```
配列オブジェクトmyArray を作成します。

```
myWeek = new Array("日","月","火","水","木","金","土");
```
配列オブジェクトmyWeek を作成し、各要素に日～土を指定します。

```
myArray[3] = "Safari";
```
配列オブジェクトmyArray の4 番目の要素をSafari にします。

```
for(i = 0; i < myArray.length; i++) {……}
```
配列オブジェクトmyArray の要素数だけ……の処理を繰り返します。

```
dogs = new Array();
dog['コウ'] = '秋田犬';
dog['ハク'] = 'サモエド';
```
犬の名前で連想配列を作成し、犬種を代入します。

```
ani = new Array();
ani.rabbit = 'うさぎ';
ani.bear = 'くま';
```
動物の種類を表す英文字列で連想配列を作成し、日本語名を代入します。

▶ ブラ[illegible]対応表	IE10	IE9	[illegible]	[illegible]	Chrome	Safari	Opera	[illegible]	[illegible]
	○	○	○	○	○	○	○	○	○

（基礎編）オブジェクトを扱う ············ P.037
【SAMPLE】2 つの配列を操作する ·········· P.176

ARRAY.02

配列要素を追加／削除したい

▲ = ★.pop()	最後尾の要素を取り出す
▲ = ★.shift()	先頭の要素を取り出す
★.push(◆)	配列の最後尾に要素を追加
★.unshift(◆)	配列の先頭に要素を追加

★……Arrayオブジェクト
▲……取り出した要素
◆……追加する要素

種類 メソッド

配列の要素を追加／削除するメソッドです。

pop、shiftメソッド

popメソッドは配列の最後尾の要素、shiftメソッドは配列の先頭の要素を取り出し、その要素を返します。取り出した要素は配列から削除します。

つまり

```
myArray = new Array("blue", "white", "red");
menu1 = myArray.pop();
```

のように配列を作成してからpopメソッドを使用した場合、配列myArrayの要素はblue、whiteの2つになり、変数menu1にはredを格納します。

push、unshiftメソッド

pushメソッドは配列の最後尾、unshiftメソッドは配列の先頭に要素◆を追加します。複数の要素を「,」(カンマ)で区切って指定すれば、一度に複数の要素を追加することも可能です。unshiftメソッドで要素を追加した際、あらかじめあった要素は1つずつ後ろにずれます。

文例

```
myArray.pop();
```

配列myArray の最後尾の要素を取り出します。

```
myArray = new Array("東京", "大阪", "京都");
menu1 =myArray.shift();
```

配列myArray の先頭の要素(東京)を取り出して、menu1に代入します。myArrayの要素は「大阪」、「京都」の2つになります。

```
myArray.push("butterfly");
```

配列myArrayの最後尾にbutterflyを追加します。

```
myArray.unshift(document.form1.nickname.value);
```

配列myArrayの先頭にform1のエレメントnicknameの値を追加します。

ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

参照

データを並べ替えたい・・・・・・・・・・・・・・・・・・・・P.172
配列要素の分割／統合／置換をしたい ・・・・・・P.174
【SAMPLE】2 つの配列を操作する ・・・・・・・・・・P.176

ARRAY.03

データを並べ替えたい

▲ = ★.sort(◆)　　配列要素を並べ替える
▲ = ★.reverse()　　配列要素を逆順に並べる

★……Arrayオブジェクト
◆……並べ替えの方法【省略可】
▲……結果を格納する変数

形式 メソッド

配列の要素を並べ替えるメソッドです。

sortメソッド

配列の要素を並べ替えます。並べ替えの方法◆を省略した場合は、文字コード順(昇順)に並べ替えられます。[illegible]であっても文字コード順になる点に注意してください。たとえば、配列(40, 100, 80, 10)の要素を並べ替えると、(10, 100, 40, 80)になります。数値を数値順に並べ替えるには、並べ替えの方法を関数で指定します(右記コラム参照)。

reverseメソッド

配列の要素の並び順を反転させます。たとえば配列(5, 3, 7, 2)の場合は(2, 7, 3, 5)になります。

文例

```
myData =myArray.sort(downFunc);
```

配列myArray の要素を関数downFunc のルールで並び替えたものを変数myData に代入します。

```
myData =myArray.reverse();
```

配列myArray の要素の並び順を反転させたものを変数myData に代入します。

ARRAY.04

配列要素の分割／統合／置換をしたい

▼ = ★.concat(◆)	2つの配列を結合
▼ = ★.slice(▲, △)	配列の要素を取り出す
★.splice(▲, ◆, ●, ●,...)	配列を置換
▼ = ★.join(■)	配列の要素を結合

▼……結果を格納する変数
★……Arrayオブジェクト
◆……結合する配列名
▲……開始位置
△……終了位置
◆……置換する文字数
●……置換データ【省略可】
■……区切り文字【省略可】

形式 メソッド

配列の要素を操作するメソッドです。要素の参照番号は0から始まります。

concatメソッド

2つの配列を結合し、1つの配列として返します。

sliceメソッド

指定した開始位置▲から終了位置△までの範囲の要素を取り出して、配列として返します。取り出されるのは終了位置の1要素前までです。

たとえば配列の3番目から5番目までの要素を取得したい場合は次のように記述します。

```
slice(2, 5)
```

spliceメソッド

指定した開始位置▲から終了位置△で指定した数分の要素を置き換えます。複数の置換データ（要素）を「,」（カンマ）で区切って指定することも可能です。指定した置換データの数によって、配列の要素数は変化します。たとえば配列の3番目から5番目までの要素を置き換えたい場合は次のように記述します。

```
splice(2, 3, "置換文字")
```

3番目だけを置き換えたい場合には次のように記述します。

```
splice(2, 1, "置換文字")
```

joinメソッド

配列の要素を指定した区切り文字■で連結し、結果の文字列を返します。区切り文字■を省略した場合は「,」（カンマ）で結合します。

文例

```
myArray =myArray1.concat(myArray2);
```
配列myArray1とmyArray2を連結し、変数myArrayに格納します。

```
myStr =myArray.slice(3, 6);
```
配列myArrayの4～6番目の要素を取り出し、配列としてmyStrに格納します。

```
myArray.splice(0, 2, "red", "green", "yellow");
```
配列myArrayの1～3番目の要素をそれぞれred、green、yellowに置き換えます。

```
document.write(myArray.join("/"));
```
配列myArrayの要素を「/」（スラッシュ）で区切って結合し、表示します。

▶ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照

配列要素を追加／削除したい・・・・・・・・・・・・・P.170
データを並べ替えたい・・・・・・・・・・・・・・・・・P.172
【SAMPLE】2つの配列を操作する・・・・・・・・・P.176

ARRAY.SAMPLE-01

2つの配列を操作する

2つの配列を操作し、表示するサンプルです。画面上のテキスト入力フィールド、セレクトボックスやボタンを使って2つの配列を操作できます。

要素の追加、置換を行うには、最上部の「要素の文字列」テキスト入力フィールドに文字列を入力します。そして、操作したい配列をセレクトボックスで選択し、「(配列)の先頭に要素を追加する」の[追加]ボタンをクリック(case0：unshift()メソッド)するか、「(配列)の(テキスト入力フィールド)番目の要素を要素の文字列に置換させる」の[置換]ボタンをクリック(case4：spliceメソッド)します。

要素の削除する際は、配列を選択し、「(配列)の最後尾の要素を削除する」の[削除]ボタンをクリック(case1：popメソッド)します。

結合する場合は、セレクトボックスで結合させたい2つの配列を選択し、[結合]ボタンをクリック(case2：concatメソッド)するか、配列の一部をテキスト入力フィールドで指定して[結合]ボタンをクリック(case3：concatメソッド、sliceメソッド)します。

操作した結果の配列の状態は下部のテキストエリアにそれぞれ表示されます。配列の要素を表示する際には、joinメソッドによって配列の要素を結合しています。joinメソッドは引数を省略すると配列の要素を「,」(カンマ)で区切ります。

JavaScript

```
var myArray = new Array();
myArray[0] = new Array();
myArray[1] = new Array();

//画面での入力にそって配列を操作する関数
function changeArray(num){
    var formElem = document.getElementById("form1");
    //選択されている項目(配列)のindex値を変数に代入
    var listn1 = formElem.s01.selectedIndex;
    var listn2 = formElem.s02.selectedIndex;
    //クリックされたボタンによって処理を振り分ける
    switch(num){
        case 0: //(先頭に要素を)「追加」ボタン
            myArray[listn1].unshift(formElem.input1.value);
            break;
        case 1: //(最後尾の要素を)「削除」ボタン
            myArray[listn1].pop();
            break;
        case 2: //(配列を)「結合」ボタン
```

```
            myArray[listn1] = myArray[listn1].concat(myArray[listn2]);
            break;
        case 3: //(要素を指定して)「結合」ボタン
            var tempArray = new Array();
            tempArray = myArray[listn2].slice(formElem.startNum.value-1,
formElem.endNum.value);
            myArray[listn1] = myArray[listn1].concat(tempArray);
            break;
        case 4: //「置換」ボタン
            myArray[listn1].splice(formElem.startNum2.value-1,1, formElem.
input1.value);
            break;
    }
    formElem.output1.value = myArray[0].join();
    formElem.output2.value = myArray[1].join();
}
```

HTML

※ボーダーは外部CSSで指定しています

```
<body>
<form action="" id="form1">
    <table>
        <tr><th>要素の文字列</th></tr>
        <tr><td><input type="text" name="input1" size="20" /></td></tr>
    </table>
    <table>
        <tr><th colspan="3">処理内容</th></tr>
        <tr>
            <td rowspan="5"><select name="s01">
            <option>配列1</option><option>配列2</option></select></td>
            <td colspan="2">の先頭に要素を
            <input type="button" value="追加" onclick="changeArray(0)" /></td>
        </tr>
        <tr>
            <td colspan="2">の最後尾の要素を
            <input type="button" value="削除" onclick="changeArray(1)" /></td>
        </tr>
        <tr>
        <td rowspan="2">に<select name="s02">
            <option>配列1</option><option>配列2</option></select></td>
            <td>を
            <input type="button" value="結合" onclick="changeArray(2)" /></td>
        </tr>
        <tr>
            <td >の<input type="text" name="startNum" size="3" />～
            <input type="text" name="endNum" size="3"/>番目の要素を
```

```
        <input type="button" value="結合" onclick="changeArray(3)" /></td>
      </tr>
      <tr>
        <td colspan="2">の
        <input type="text" name="startNum2" size="3" />番目の要素を要素の文字列に
        <input type="button" value="置換" onclick="changeArray(4)" /></td>
      </tr>
    </table>
    <table>
      <tr><th>配列1</th><th>配列2</th></tr>
      <tr><td><textarea name="output1" rows="6" cols="20" disabled="disabled"></textarea></td>
        <td><textarea name="output2" rows="6" cols="20" disabled="disabled"></textarea></td></tr>
    </table>
</form>
</body>
```

[追加]ボタンで配列1と配列2に要素を配置した状態

配列1に配列2のすべての要素を結合します

配置1に配列2の2～3番目の要素を結合します

配列1の2番目の要素pinkを新たな要素yellowに置換します

参照

Array オブジェクト …… P.168
pop メソッド …… P.170
unshift メソッド …… P.170
concat メソッド …… P.174
slice メソッド …… P.174
splice メソッド …… P.174
join メソッド …… P.174

日付や時刻を扱いたい

★ = new Date(◆)　日付のオブジェクトを作成

★……新しい日付のオブジェクト
◆……設定する日時【省略可】

形式　オブジェクト

日付や時刻を扱うには、newステートメント(p.039)を使用して、Date(日付)オブジェクトを作成します。Dateオブジェクトを使うと。現在の年月日や時刻を表示したり。指定した日付までの日数を調べたりすることができます。

特定の日付や時刻を持ったDateオブジェクトを作成する場合は日付を表す文字列("1987/1/4","Jan 4,1987"など）を引数として◆ に指定するのが一般的です。この文字列はparseメソッド(p.200)で認識可能な日時を表すものです。そのほかの方法として、1970年1月1日00:00:00(UTC)からのミリ秒を表す整数値を引数とする方法、4桁の年、月(0～11)、日(1～31)、時(0～23)、分(0～59)、秒(0～59)、ミリ秒(0～999)を「,」(カンマ）で区切ったyear, month, day, [hours, minutes, seconds, ms]([] 内は省略可)の形式を引数とする方法があります。

なお、引数を省略すると、現在の日付と時刻に設定されます。

文例

```
day = new Date(1998, 11, 31, 1, 2, 3, 456);
```

1998年12月31日1時2分3秒456ミリ秒を値とするDateオブジェクトdayを作成します。

```
day = new Date(915030000000);
```

915030000000を値とするDateオブジェクトdayを作成します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照
(基礎編) オブジェクトを扱う ············ P.037
【SAMPLE】カレンダーを作成する ········· P.194

日付を設定したい

★.setFullYear(◆) 4桁の西暦を設定
★.setYear(▲) 西暦を設定
★.setMonth(●) 月を設定
★.setDate(■) 日を設定

★……Dateオブジェクト
◆……4桁の西暦年
▲……西暦年
●……月(0～11)
■……日(1～31)

形式 メソッド

Dateオブジェクトの日付を設定するメソッドです。

setFullYearメソッド

4桁の西暦を設定します。

setYearメソッド

西暦を設定します。getYearメソッド(p.196)と同様にブラウザの種類や機種によって結果が異なる場合があるので、setFullYearメソッドを使用した方がよいでしょう。

setMonthメソッド

月を設定します。設定する際は実際より1小さい値を指定します(1月=0、……、12月=11)。

setDateメソッド

日を設定します。

文例

```
myDate.setFullYear(2008);
```
年を2008年に設定します。

```
myDate.setDate(document.form1.d.value);
```
日をform1のdの値に設定します。

▶ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

参照

日付を取得したい…………P.182
時刻を設定したい…………P.184
時刻を取得したい…………P.185
【SAMPLE】カレンダーを作成する…………P.194

DATE.03

日付を取得したい

◆ = ★.getFullYear()	4桁の西暦を返す
◆ = ★.getYear()	西暦を返す
◆ = ★.getMonth()	月を返す(0～11)
◆ = ★.getDate()	日を返す(1～31)
◆ = ★.getDay()	曜日を返す(0:日～6:土)

◆……結果を格納する変数
★……Dateオブジェクト

式 メソッド

Dateオブジェクトの日付や曜日を参照するメソッドです。

getFullYearメソッド

4桁の西暦を返します。

getYearメソッド

西暦から1900を引いた値(1977年なら77)を返します。ただし、2000年を超えた場合はブラウザや機種によって4桁の値(2000年なら2000)を返すものと、1900を引いた値(2000年なら100)を返すものがあります。古いブラウザでは対応していませんが、年を扱う際には4桁の西暦を返すgetFullYearメソッドを使用した方がよいでしょう。

etMonthメソッド

月を返します。getMonthメソッドで返される値は、実際の月よりも1小さい値(1月=0、2月=1、3月=2、……、12月=11)となることに注意してください。

getDateメソッド

日を返します。

getDayメソッド

曜日を返します。getDayメソッドで返される曜日は0から6の範囲の値(日曜=0、月曜=1、……、土曜=6)になります。曜日を表示する際には、曜日の文字列をあらかじめ配列に設定し、

配列のインデックス番号として曜日の値を指定するのが一般的です。

文例

```
theFyear = now.getFullYear();
```
現在の西暦を4桁で取得し、変数theFYear に代入します。

```
theYear = now.getYear();
```
現在の西暦を取得し、変数theYear に代入します。

```
document.write("今は" + (now.getMonth()+1) + "月です");
```
現在の月を表示します。

```
alert("今日は" + now.getDate() + "日。");
```
現在の日をダイアログに表示します。

```
theDate =now.getDay();
```
現在の曜日の数値を取得し、変数theDate に代入します。

```
theWeek = new Array("日" ,"月" ,"火" ,"水" ,"木" ,"金" ,"土");
theDate = theWeek [now.getDay()];
```
現在の曜日の数値を取得し、配列theWeek から該当する曜日を取得して変数theDateに代入します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	[illegible]	Android
	○	○	○	○	○	○	○	○	○

参照

[illegible]更新日を自動的に挿入したい・・・・・・・・・・P.059
日付を設定したい・・・・・・・・・・・・・・・・・・・・P.181
時刻を取得したい・・・・・・・・・・・・・・・・・・・・P.185
時刻を設定したい・・・・・・・・・・・・・・・・・・・・P.184
さまざまな形式で日付を表示したい・・・・・・・・P.188
【SAMPLE】カレンダーを作成する・・・・・・・・・P.194

DATE.04

時刻を設定したい

★.setHours(◆) [illegible]を設定
★.setMinutes(▲) 秒を設定
★.setSeconds(●) 分を設定
★.setMilliseconds(■) ミリ秒を設定

★……Dateオブジェクト
◆……時(0～23)
▲……分(0～59)
●……秒(0～59)
■……ミリ秒(0～999)

形式 メソッド

Dateオブジェクトの[illegible]刻を設定します。引数には任意の時、分、秒、ミリ秒(1000分の1秒)を[illegible]または数式で設定します。これらのメソッドで設定した時刻はシステム(PC)の時計には影響を与えません。

文例

```
myTime.setHours(23);
```
時刻(時)を23時に設定します。

```
myTime.setMinutes(23);
```
時刻(分)を23分に設定します。

```
myTime.setSeconds(23);
```
時刻(秒)を23秒に設定します。

```
myTime.setMilliseconds(23);
```
時刻(ミリ秒)を0.023秒に設定します。

▶ [illegible]対応表	IE10	[illegible]	IE8	[illegible]	[illegible]	Safari	Opera	[illegible]	Android
	○	○	○	○	○	○	○	○	○

参照
日付を設定したい………………P.181
日付を取得したい………………P.182
時刻を取得したい………………P.185

DATE.05

時刻を取得したい

```
◆ = ★.getHours()          時を返す(0～23)
◆ = ★.getMinutes()        分を返す(0～59)
◆ = ★.getSeconds()        秒を返す(0～59)
◆ = ★.getMilliseconds()   ミリ秒を返す(0～999)
```

◆……結果を格納する変数
★……Dateオブジェクト

形式 メソッド

現在の時刻を求めるメソッドです。getMillisecondsメソッドはミリ秒の値を0～999の数値で返します。

文例

```
document.write("今、" +now.getHours()+ "時" +now.getMinutes() + "分です。");
```

現在時刻(時と分)を書き出します。

```
s = now.getSeconds();
```

現在時刻(秒)を取得し、変数sに代入します。

```
ms = now.getMilliSeconds();
```

現在時刻(ミリ秒)を取得し、変数msに代入します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

参照

最終更新日を自動的に挿入したい…………P.059
日付を設定したい……………………P.181
日付を取得したい……………………P.182
時刻を設定したい……………………P.184
さまざまな形式で日付を表示したい………P.188
【SAMPLE】来年までの時間をカウントダウンする‥P.196

DATE.06

指定した時間までの経過秒数を求めたい

○ = ★.getTime()　現在時刻までのミリ秒を返す
○ = Date.parse(◆)　指定日時までのミリ秒を返す
○ = Date.UTC(▲, ●, ■, ▼, ☆, ◇)　指定日時までのミリ秒を返す

○……結果を格納する変数
★……Dateオブジェクト
◆……日付文字列("1977/1/4"など)
▲……4桁の西暦(1970より大きい値)
●……月(0～11)
■……日(1～31)
▼……時(0～23)
☆……分(0～59)
◇……秒(0～59)

形式 メソッド

いずれも1970年1月1日午前0時からの経過秒数をミリ秒で返すメソッドですが、対象となる日付の指定方法が異なります。

getTimeメソッド

Dateオブジェクトを対象とします。

parseメソッド

日付を表す文字列("1977/1/4"、"Jan 4, 1997"など)で指定した日時を対象とします。

UTCメソッド

4桁の西暦、月、日、時、分、秒をそれぞれ「,」(カンマ)で区切って指定した日時を対象とします。月を設定する際は、実際より1小さい値(1月=0、2月=1、……、12月=11)を指定します。

文例

```
ms = today.getTime();
```

1970年1月1日午前0時から現在時刻までの経過秒数をミリ秒で取得し、msに代入します。

```
time = Date.Parse("2010/8/31");
```

1970年1月1日午前0時から2010年8月31日までの経過秒数をミリ秒で取得し、変数timeに代入します。

```
time = Date.UTC(2010, 7, 31, 23, 59, 59);
```

1970年1月1日午前0時から2010年8月31日23時59分59秒までの経過秒数をミリ秒で取得し、変数timeに代入します。

ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

参照
【SAMPLE】来年までの時間をカウントダウンする・・P.196

さまざまな形式で日付を表示したい

```
◆ = ★.toGMTString()      グリニッジ標準時で返す
◆ = ★.toLocaleString()   ローカル時で返す
◆ = ★.toUTCString()      協定世界時で返す
```

★……Dateオブジェクト
◆……結果を格納する変数

形式 メソッド

★で指定されたDateオブジェクトをさまざまな形の日付文字列で返します。これらのメソッドは、ブラウザやそのバージョン、およびシステム(PC)の設定によって返される文字列の形式が異なります。なお、ローカル時とはシステム(PC)に設定されている地域の時刻です。

文例

```
window.status("グリニッジ標準時：" + now.toGMTString());
```
ステータスバーにグリニッジ標準時を表示します。

```
lstr = now.toLocaleString();
```
ローカル時を変数lstrに代入します。

```
document.getElementById("output").value = now.toUTCString();
```
協定世界時をID名outputの値にします。

Column 協定世界時とは

全世界で時刻を記録する際に使われる公式な時刻です。天体観測を元に定められるGMT(グリニッジ標準時)とほぼ同じですが、セシウム原子時計で計測した時間(国際原子時)とGMTの差が0.9秒以内になるよう人工的に調整されており、世界共通の標準時となっています。

▶ [illegible]	IE10	IE9	[illegible]	[illegible]	[illegible]	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照

日付を取得したい……P.182
時刻を取得したい……P.185
協定世界時で表示したい……P.190
【SAMPLE】来年までの時間をカウントダウンする……P.196

DATE.08

協定世界時で設定したい

★.setUTCFullYear(◆)	協定世界時の4桁の西暦で設定
★.setUTCMonth(▲)	協定世界時の月で設定
★.setUTCDate(●)	協定世界時の日で設定
★.setUTCHours(■)	協定世界時の時で設定
★.setUTCMinutes(▼)	協定世界時の分で設定
★.setUTCSeconds(☆)	協定世界時の秒で設定
★.setUTCMilliseconds(◇)	協定世界時のミリ秒で設定

★……Dateオブジェクト
◆……4桁の西暦年
▲……月(0～11)
●……日(1～31)
■……時(0～23)
▼……分(0～59)
☆……秒(0～59)
◇……ミリ秒(0～999)

形式 メソッド

協定世界時で設定するメソッドです。setUTCMonthメソッドでは、設定する月には実際より1小さい値を設定します(1月=0、2月=1、……、12月=11)。

文例

```
myDate.setUTCFullYear(2013);
```
協定世界時の年を2013 年に設定します。
```
myDate.setUTCMonth(11);
```
協定世界時の月を12 月に設定します。
```
myDate.setUTCDate(document.form1.d.value );
```
協定世界時の日をform1 のd の値に設定します。
```
myDate.setUTCHours(23);
```
協定世界時の時を23 時に設定します。
```
myDate.setUTCMilliseconds(23);
```
協定世界時のミリ秒を0.023 秒に設定します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOSS	Android
	○	○	○	○	○	○	○	○	○

参照
さまざまな形式で日付を表示したい ········ P.188
協定世界時で表示したい ················ P.190

DATE.09

協定世界時で表示したい

◆ = ★.getUTCFullYear()	協定世界時の4桁の西暦で返す
◆ = ★.getUTCMonth()	協定世界時の月で返す
◆ = ★.getUTCDate()	協定世界時の日で返す
◆ = ★.getUTCDay()	協定世界時の曜日で返す
◆ = ★.getUTCHours()	協定世界時の時で返す
◆ = ★.getUTCMinutes()	協定世界時の分で返す
◆ = ★.getUTCSeconds()	協定世界時の秒で返す
◆ = ★.getUTCMilliseconds()	協定世界時のミリ秒で返す
◆ = ★.getTimezoneOff set()	協定世界時との時差で返す

★……Dateオブジェクト
◆……結果を格納する変数

形式 メソッド

協定世界時の西暦や月などを調べるメソッドです。

getUTCMonthメソッド

協定世界時の月で返します。ただし、返される値は実際の月よりも1小さい値(1月=0、2月=1、……、12月=11)となることに注意してください。

getUTCDayメソッド

協定世界時の曜日で返します。返される曜日は0から6の範囲の値(日曜=0、月曜=1、……、土曜=6)になります。曜日を表示する際には、曜日の文字列をあらかじめ配列に設定し、配列のインデックス番号として曜日の値を指定するのが一般的です。

getTimezoneOff setメソッド

協定世界時との時差を分単位で返します。ただし。ブラウザやOSの種類によっては正確な値が返されない場合があります。

文例

```
fy = today.getUTCFullYear();
m = today.getUTCMonth()+1;
d = today.getUTCDate();
document.write(fy + "年" + m + "月" + d + "日");
```

現在の協定世界時の西暦、月、日をそれぞれ変数fy、m、dに代入し、表示します。

```
theWeek = new Array("日","月","火","水","木","金","土");
theDate = theWeek [now.getUTCDay()];
```

現在の協定世界時における曜日の数値を取得し、配列theWeek から該当する曜日を取得して変数theDateに代入します。

```
h =today.getUTCHours();
m = today.getUTCMinutes();
s = today.getUTCSeconds();
ms = today.getUTCMilliseconds();
ofst = today.getTimezoneOffset();
alert(h + "時" + m + "分" + s + "秒" + ms + ":時差" + ofst + "分");
```

現在の協定世界時の時、分、秒、ミリ秒、時差をそれぞれ変数に代入し、ダイアログに表示します。

▶ [illegible]ウザ対応表	IE10	IE9	[illegible]	Fx	Chrome	Safari	[illegible]	iOS6	Android
	○	○	○	○	○	○	○	○	○

さまざまな形式で日付を表示したい ・・・・・・・・ P.188
協定世界時で設定したい ・・・・・・・・・・・・・・・ P.189

DATE.SAMPLE-01

さまざまな形式で現在時刻を表示する

さまざまな形式で現在時刻を表示するサンプルです。ページ読み込み時にタイマーをセットし、1秒ごとに表示を更新します。

JavaScript

```
//様々な形式で日時を表示する
function showTime(){
    var formElem = document.getElementById("form1");
    today = new Date();
    formElem.output1.value = today.toGMTString();
    formElem.output2.value = today.toUTCString();
    formElem.output3.value = today.toLocaleString();
}
```

HTML

```
<body id="body" onload="setInterval('showTime()',1000)"><!--onload属性にタイマーをセット-->
  <form action="" id="form1">
    <p>
    グリニッジ標準時：<input type="text" name="output1" size="35" /><br />
    協定世界時(UTC)：<input type="text" name="output2" size="35" /><br />
    ローカル時(地域)：<input type="text" name="output3" size="35" />
    </p>
  </form>
</body>s
```

Internet Explorer

各種形式で現在の時刻を1秒ごとに表示します(ブラウザによって形式が異なります)

参照
toGMTString メソッド ･････････････････ P.188
toLocaleString メソッド ････････････････ P.188
toUTCString メソッド ･･････････････････ P.188

カレンダーを作成する

カレンダーを作成するサンプルです。まず、現在の日付を取得し、年と月を参照します。各月の日数は、日に31までの値を設定して日付オブジェクトを作成し、月が変わらない値を見つけることで確認できます。たとえば「new Date("2008","1","32") 」(2008年1月32日)と指定して日付オブジェクトを作成すると2008年2月1日となります。

さらに、取得した情報を元に変数htmlにHTMLとその内容を付け足していきます。最後に作成したHTMLをinnerHTMLで<body>タグの中に挿入して完成です。

JavaScript

```
var year; //変数の宣言 年
var mon; //月
var maxDate; //日の上限
var dayArray = new Array("日","月","火","水","木","金","土"); //曜日の配列
var code = ""; //カレンダー作成のためのhtmlコードを代入していく変数

//今月のカレンダーを作成する関数
function calendar(){
    var nowDate = new Date(); //現在の日付を取得
    year = nowDate.getFullYear(); //4桁の年を取得
    mon = nowDate.getMonth(); //月を取得
    for (var i = 27; i <= 31; i++){   //今月の日の上限を取得する
        var check = new Date(year,mon,i); //27から31までの数字をセットして日付を作成
        if (check.getMonth() == mon){
                        //変数checkが日付として正しければ変数maxDateに変数iの値を代入
            maxDate = i;
        }else{
            break;
        }
    }
    code += "<p><b>" + year + "年  " + (mon+1) + "月</b></p>";
    code += "<table><tr>";
    for(var i = 0; i < dayArray.length; i++){    //曜日の見出し部分のhtml作成
        code += "<td>" + dayArray[i] + "</td>";
    }
    code += "</tr><tr>";
    var date = new Date(year,mon,1); //今月の1日の日付を取得
    for(var i = 0; i < date.getDay(); i++){    //1日の曜日までの空欄部分を作成
        code += "<td></td>";
```

```
    }
    for(var i = 1; i <= maxDate; i++){    //日にち(1～maxDate)部分の作成
        if(date.getDay() == 0 && i != 1){    //日にちの曜日が日曜日になったら次の行へ
            code += "</tr><tr>";
        }
        code += "<td>"+ i +"</td>"; //日にち部分作成
        date.setDate(date.getDate() + 1); //変数dateの日付を1日進める
        if(i == maxDate){    //最終日以降までの空欄部分を作成
            while(date.getDay() != 0){
                code += "<td></td>";
                date.setDate(date.getDate() + 1);
            }
        }
    }
    code += "</tr></table>";
    document.getElementById("body").innerHTML = code;
                                          //bodyタグ内に作成したhtmlを挿入
}
```

HTML

```
<body id="body" onload="calendar()">
</body>
```

Internet Explorer

その月のカレンダーが表示されます

参照
Date オブジェクト P.180
getMonth メソッド P.182
getDate メソッド P.182
getDay メソッド P.182
setFullYear メソッド P.181
setDate メソッド P.181

DATE.SAMPLE-03

来年までの時間をカウントダウンする

来年までの時間をカウントダウンするサンプルです。ページが読み込まれると<body>タグのonload属性に指定されたタイマーをセットし、1秒ごとにcountDown関数を呼び出します。この関数は現在の時間から指定時間(来年の1月1日0時0分0秒)までの時間を計算します。Date.UTC(指定時間)-Date.UTC(現在時間)で指定時間までの時間をミリ秒単位で取得し、取得した値からそれぞれ日数、時間、分、秒を割り出して表示します。1秒ごとに更新されるのでカウントダウンのようになります。

JavaScript

```
//来年までの時間を計算する関数
function countDown(){
    var today = new Date(); //現在時間を取得
    var year = today.getFullYear();    //年、月、日、時間、分、秒をそれぞれ取得
    var month = today.getMonth();
    var date = today.getDate();
    var hour = today.getHours();
    var minute = today.getMinutes();
    var second = today.getSeconds();
    var mSec = Date.UTC((year + 1), 0, 1, 0, 0, 0) - Date.UTC(year, month, date,
hour, minute, second); //来年までのミリ秒数を取得
    var sec = mSec / 1000;    //取得したミリ秒数から日数、時間、分、秒を割り出す
    var s = sec % 60; //秒
    var min = (sec - s) / 60;
    var m = min % 60;
    var hou = (min - m) / 60;
    var h = hou % 60; //時間
    var d = (hou - h) / 24; //日数
    document.getElementById("date").value = d;    //画面へ表示
    document.getElementById("hour").value = h;
    document.getElementById("minute").value = m;
    document.getElementById("second").value = s;
}
```

HTML

```
<body onload="setInterval('countDown()',1000)">
  <form action="" id="">
    <p>
    来年まであと
    <input type="text" id="date" size="3" />日
    <input type="text" id="hour" size="3" />時間
    <input type="text" id="minute" size="3" />分
    <input type="text" id="second" size="3" />秒です。
    </p>
  </form>
</body>
```

Internet Explorer

来年までの時間がカウントダウンされます

参照
getHours メソッド・・・・・・・・・・・・・・・・・・・P.185
getMinutes メソッド・・・・・・・・・・・・・・・・・P.185
getSeconds メソッド・・・・・・・・・・・・・・・・・P.185
Date.UTC メソッド・・・・・・・・・・・・・・・・・・P.186

STRING.01

文字列を扱いたい

★ = new String(◆)　　Stringオブジェクトを作成
★.length　　文字列の長さを参照

★……Stringオブジェクト名
◆……文字列

形式　オブジェクト(String)、プロパティ(length)

文字列(Stringオブジェクト)は文字列を扱うオブジェクトです。色やサイズの変更、各種の修飾、文字の検索や抜き出しなど、文字列を操作できます。

Stringオブジェクト

Stringオブジェクトを明示的に作成するには、newステートメント(p.037)を使用します。ただし、このようにnewステートメントを使用してStringオブジェクトを作成することは少なく、通常は変数の値に文字列を代入するなどで作成します。文字列は「""」(ダブルクォーテーション)または「''」(シングルクォーテーション)で囲んで記述します。

lengthプロパティ

文字列の長さを調べます。

文例

```
str = new String("Hello!");
```
文字列Hello!をStringオブジェクトstrとして作成します。str = "Hello";と記述しても同じです。

```
str = new String(2007);
```
数値2007をStringオブジェクトstrとして作成します。

```
myStr = str.length;
```
Stringオブジェクトstrの長さを調べて変数myStrに代入します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	[illegible]	iOS6	Android
	○	○	○	○	○	○	○	○	○

(基礎編) オブジェクトを扱う ・・・・・・・・・・・・ P.037

STRING.02

文字列にリンクやアンカーを設定したい

● = ★.link(◆)　リンクを設定

● = ★.anchor(▲)　アンカー名を設定

●……HTML/XHTMLタグが付加された文字列
★……Stringオブジェクト
◆……URI
▲……アンカー名

形式　メソッド

指定した文字列にリンクやアンカー名を設定するメソッドです。これらのメソッドでは、結果として以下のようにHTML/XHTMLタグが付加された文字列を返します。

メソッドの指定	メソッドによって返される文字列
文字列.link("http://www.ank.co.jp/")	文字列
文字列.anchor("top")	文字列

文例

```
document.write("株式会社翔泳社".link("http://www.shoeisha.co.jp/"));
```

文字列「株式会社翔泳社」にリンクを設定して表示します。

```
document.write("CONTENTS".anchor("contents"));
```

文字列CONTENTS にアンカー名contents を設定して表示します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照

大文字／小文字に変換したい・・・・・・・・・・・・・P.200

STRING.03

大文字／小文字に変換したい

```
● = ★.toUpperCase()    大文字に変換
● = ★.toLowerCase()    小文字に変換
```

●……変換された文字列を格納する変数
★……Stringオブジェクト

形式 メソッド

指定した文字列に含まれるアルファベットを、大文字または小文字に揃えます。

通常、indexOfメソッド（p.202）などで検索を行うときには大文字と小文字が区別されます。また、sortメソッド（p.172）での並べ替えは文字コード順になるので、小文字より大文字の方が先になってしまい、c、B、a、d、Eを並べ替えると、B、E、a、c、dになってしまいます。検索や並べ替えを行う際に大文字と小文字を区別しない場合には、あらかじめtoUpperCaseメソッドやtoLowerCaseメソッドでどちらかに揃えてから実行するとよいでしょう。

文例

```
alert(word1.toUpperCase());
```

Stringオブジェクトword1の値を大文字に変換し、ダイアログに表示します。

```
str = word2.toLowerCase();
```

Stringオブジェクトword2の値を小文字に変換し、変数strに代入します。

▶ ブラウザ対応表

IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
○	○	○	○	○	○	○	○	○

参照

文字列にリンクやアンカーを設定したい …… P.199
データを並べ替えたい …… P.172
文字列を検索したい …… P.202

STRING.04

文字列を分割したい

● = ★.split(▲)

●……結果を格納する配列
★……Stringオブジェクト
▲………区切り文字

形式 メソッド

文字列を指定した区切り文字▲で分割し、結果を配列として返すメソッドです。区切り文字は「,」(カンマ)である必要はなく、「/」(スラッシュ)や「" "」(空白)などにすることも可能です。

文例

```
girl = "Alice/Dorothy/Jessica/Mary/Shelley/Vivian".split("/");
```

文字列「Alice/Dorothy/Jessica/Mary/Shelley/Vivian」を「/」(スラッシュ)の部分で区切り、配列girlに格納します。

```
data = member.split(" ");
```

配列memberの要素を「" "」(空白)で区切り、配列dataに格納します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

参照
文字を抜き出したい …………………… P.204
文字列の結合や抜き出しを行いたい ……… P.205

STRING.05

文字列を検索したい

● = ★.indexOf(◆, ▲)　　文字列を検索
● = ★.lastIndexOf(◆, ▲)　　文字列を後ろから検索

●……見つかった文字列の位置を格納する変数
★……Stringオブジェクト
◆……検索文字列
▲……検索開始位置(数値)【省略可】

形式 メソッド

文字列を検索するメソッドで、大文字／小文字を区別して検索できます。大文字／小文字が混在している場合は、対象となる文字列をあらかじめtoUpperCase、toLowerCaseメソッド(p.200)で大文字または小文字に揃えておくとよいでしょう。

indexOfメソッド

文字列の左側から指定した文字列◆を検索し、最初に一致した文字列の先頭位置を返します。一致する文字列がなかった場合は-1を返します。検索開始位置▲を指定した場合は指定した位置より後ろの部分の文字列を検索し、省略した場合は先頭から検索します。先頭の位置は0です。

lastIndexOfメソッド

文字列の最後(右側)から指定した文字列◆を検索し、最初に一致した文字列の先頭位置を返します。検索開始位置▲を指定した場合は指定した位置より前の部分の文字列に対して検索し、省略した場合は最後から検索します。

文例

```
n1 = "gewurztraminar".indexOf("urz");
```
文字列gewurztraminarの左側から文字列urzの位置を検索し、n1に代入します(n1の値は3)。

```
n2 = "gewurztraminar".indexOf("urz",2);
```
文字列gewurztraminarの左の3文字目から文字列urzの位置を検索し、n2に代入(n2の値は3)。

```
n3 = "gewurztraminar".lastIndexOf("urz");
```
文字列gewurztraminarの右側から文字列urzの位置を検索し、変数n3に代入します(n3の値は3)。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

参照

文字列を分割したい …… P.220
文字を抜き出したい …… P.204
文字列の結合や抜き出しを行いたい …… P.205
【SAMPLE】文字列を検索する …… P.207

STRING.06

文字コードを扱いたい

● = ★.charCodeAt(◆)　文字をUnicodeの値に変換

■ = String.fromCharCode(▲, ▲, ..., ▲)　Unicodeの値を文字に変換

●……Unicodeの値を格納する変数
★……対象となる文字列
◆……文字の位置
■……Unicodeの値から得られた文字列を格納する変数
▲……文字コード

形式　メソッド

文字をUnicodeの値に変換、またはその逆を行うメソッドです。

charCodeAtメソッド

文字列中の指定した位置にある文字のUnicodeの値を返します。先頭の文字の参照番号は0です。全角文字も調べることができます。

fromCharCodeメソッド

引数に与えたUnicodeの値▲を文字に変換します。「,」(カンマ)で区切って記述することにより、複数の文字コードを文字列に変換できます。

文例

```
document.write("shiraz".charCodeAt(2));
```

文字列shirazの3番目の文字iをUnicodeの値に変換して書き出します。(結果は105)。

```
myChar = String.fromCharCode(keyNum);
```

変数keyNumに格納されているUnicodeの値を文字に変換して、変数myCharに代入します。

Column　escape、unescapeメソッドとの違い

文字と文字コードの変換についてはescape、unescapeメソッドがありますが(p.250)、これらは主にクッキーやフォーム送信の文字列中の特殊文字や漢字の変換に用いるのに対し、charCodeAt、fromCharCodeメソッドは、主にキーボードの入力値を調べる際に使用します。charCodeAt、fromCharCodeメソッドは1文字だけの変換、アルファベットも変換できるといった特徴があります(escapeメソッドではアルファベットや数値は変換されません)。

ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

文字列をエンコード／デコードしたい ……P.250

STRING.07

文字を抜き出したい

● = ★.charAt(◆)

●……抜き出された文字列を格納する変数
★……Stringオブジェクト
◆……文字の位置

形式 メソッド

文字列から指定した位置の1文字を抜き出すメソッドです。先頭の文字の位置は0です。文字列から順番に1文字ずつ文字を取り出したい場合などに利用します。

文例

```
w = "sangiovese";
ws =w.charAt(3);
```

文字列sangioveseから4番目の文字を抜き出して、変数wsに代入します(wsの値はg)。

```
str = "chardonnay".charAt(3);
```

文字列chardonnayから4番目の文字を抜き出して、変数strに代入します(strの値はr)。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照
文字列を検索したい ……………… P.202
文字列の結合や抜き出しを行いたい ……… P.205

STRING.08

文字列の結合や抜き出しを行いたい

● = ★.concat(◆)　文字列を結合
● = ★.slice(▲, △)　文字列を範囲で抜き出す
● = ★.substring(▲, △)　文字列を範囲で抜き出す
● = ★.substr(▲, ■)　文字列を文字数で抜き出す

●……結果として得られる文字列を格納する変数
★……Stringオブジェクト
◆……結合する文字列
▲……開始位置
△……終了位置
■……抜き出す文字数

形式　メソッド

文字列を結合したり、文字列を抜き出したりするメソッドです。

concatメソッド

2つの文字列(★と◆)を結合します。

slice、substringメソッド

引数に開始位置と終了位置を指定して、その範囲内の文字列を抜き出します。sliceメソッドでは終了位置に負の数を使用し、末尾から数えた位置を指定することも可能です。substringメソッドでは△と▲の順番を変更しても結果は同じになります。つまり、終了位置が開始位置より小さい場合、開始位置から前方に向かって抜き出すわけです。いずれの場合も先頭の文字の位置は0になります。また、終了位置の直前の文字までを抜き出します。

substrメソッド

開始位置から指定した文字数分の文字列を抜き出します。先頭の文字の位置は0になります。

たとえば以下の例では、どれも文字列Scrが返されます。

メソッドの指定	意味
JavaScript.slice(4, 7)	5文字目から7文字目まで
JavaScript.slice(4, -3)	5文字目から後ろから4文字目まで
JavaScript.substring(4, 7)	5文字目から7文字目まで
JavaScript.substring(7, 4)	7文字目から5文字目まで
JavaScript.substr(4, 3)	5文字目から3文字分

0 1 2 3 4 5 6 7 8 9
J a v a S c r i p t

文例

```
msg = msg1.concat(msg2);
```
文字列msg1と文字列msg2を結合して変数msgに代入します。

```
str = "grenache".slice(2, 6)
```
文字列grenacheの3文字目から6文字目までを抜き出して、変数strに代入します(strの値はenac)。

```
document.write("grenache".slice(2, -2));
```
文字列grenacheの3文字目、後ろから3文字目まで抜き出して表示します(結果はenac)。

```
str = "grenache".substring(2, 5);
```
文字列grenacheの3文字目から5文字目まで抜き出して、変数strに代入します(値はena)。

```
str = "grenache".substr(2, 3);
```
文字列grenacheの3文字目から3文字抜き出して、変数strに代入します(値はena)。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

参照
文字列を検索したい ・・・・・・・・・・・・・・・・・・ P.202
文字を抜き出したい ・・・・・・・・・・・・・・・・・・ P.204

STRING.SAMPLE-01

文字列を検索する

検索したい文字を指定し、それがテキスト入力欄の文章にいくつ含まれているかをカウントするサンプルです。[検索開始]ボタンがクリックされるとsearchStr関数を呼び出します。検索文字列および検索対象文字列をそれぞれ変数に代入し、while文の中でindexOfメソッドで繰り返し検索しています。最後まで検索が終了すると、検索開始位置用の変数posに-1を代入し、繰り返し処理を終了します。最後に検索結果をダイアログに表示します。

JavaScript

```
//文字列に指定した文字列が含まれているか検索する関数
function checkStr(){
    //変数の宣言
    var pos = 0;  //検索開始位置
    var times = 0; //検出回数
    var searchFor = document.getElementById("searchFor").value; //検索する文字

    var searchFrom = document.getElementById("searchFrom").value; //検索対象

    //検索文字列が入力されていない場合
    if(searchFor == ""){
        alert("検索する文字列を入力して下さい");
        return;
    }
    //検索対象文字列が入力されていない場合
    if(searchFrom == ""){
        alert("検索対象となる文字列がありません");
        return;
    }
    while(pos >= 0){ //検索対象を最後まで検索したらループを抜ける
        pos = searchFrom.indexOf(searchFor, pos);
        //一致する文字列が見つかった場合
        if(pos > 0){
            times++; //検出回数を+1する
            pos++; //次回の検索開始位置
        }
    }
    //検索結果をダイアログで表示
    alert("文章内から「" + searchFor + "」は" + times + "回検出されました");
}
```

HTML

※フォント・サイズは外部CSSで指定しています

```
<body>
<form action="">
  <p>
    検索する文字列：<input type="text" id="searchFor" size="28"/>
    <input type="button" name="b1" value="検索開始" onclick="checkStr()"
/><br />
   検索対象となる文字列(編集可能)：<br />
   <textarea id="searchFrom" cols="80" rows="10">
     この話はすべて遠野の人佐々木鏡石君より聞きたり。昨明治四十二年の二月ごろより始めて
夜分おりおり訪ね来たりこの話をせられしを筆記せしなり。鏡石君は話上手にはあらざれども誠実
なる人なり。自分もまた一字一句をも加減せず感じたるままを書きたり。
    (…中略…)
    以上は自分が遠野郷にてえたる印象なり。
  </textarea>
  </p>
</form>
</body>
```

指定した文字列の検出回数が表示されます

参照

indexOf メソッド ・・・・・・・・・・・・・・・・・・・・ P.202

NAVIGATOR.01

ブラウザを判別したい

navigator.appName　　アプリケーション名を参照
navigator.appVersion　　バージョンを参照

形式 プロパティ

Webブラウザの種類(名前)とバージョンといったブラウザ情報を参照できます。

appNameプロパティ

ブラウザの種類(名前)を参照するプロパティです。たとえば、Internet ExplorerはMicrosoft Internet Explorer、OperaはOpera、FirefoxとChromeはNetscapeを返します。

appVersionプロパティ

ブラウザのバージョンを参照するプロパティです。通常はバージョン番号の後ろにOS名やCPUの種類などの情報が追加されますが、返す値はブラウザや環境によって異なります。

文例

```
if(navigator.appName == "Microsoft Internet Explorer"){
    alert("MSIEをお使いですね？ ");
}
```

ブラウザの名前がMicrosoft Internet Explorer の場合、「MSIE をお使いですね？」というダイアログを表示します。

```
document.write("バージョン：", navigator.appVersion);
```

ブラウザのバージョンを表示します。

Column　Navigatorオブジェクト

NavigatorオブジェクトはWebブラウザやOSに関する情報を持つオブジェクトです。Webブラウザの種類やバージョン、設定など、取得した情報をもとに処理を変更したり、ブラウザやバージョンごとに対応するページに移動して処理を行ったりする場合などに使用できます。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

【SAMPLE】ブラウザの情報を調べる ・・・・・・・ P.216

ブラウザの情報を調べたい

navigator.appCodeName コード名を参照
navigator.browserLanguage 使用言語を参照
navigator.language 使用言語を参照
navigator.platform プラットフォームを参照
navigator.userAgent エージェント名を参照

形式 プロパティ

それぞれ、ブラウザのコード名、使用言語、エージェントなどブラウザに関する情報を参照するプロパティです。browserLanguageおよびlanguageプロパティはいずれもブラウザの使用言語を参照しますが、前者はInternet Explorer、後者はFirefoxに対応しています。

文例

```
alert(navigator.appCodeName);
```
ダイアログにブラウザのコード名を表示します。

```
document.write("使用言語：" + navigator.language);
```
ブラウザの使用言語を表示します。

```
alert("プラットフォームは：" + navigator.platform + "です。");
```
ダイアログにユーザーのプラットフォームを表示します。

```
document.write("ユーザーエージェント：" + navigator.userAgent);
```
ダイアログにブラウザのユーザーエージェント名を表示します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
browserLanguage	○	○	○	×	×	×	○	×	×
language	×	×	×	○	○	○	○	○	○
その他	○	○	○	○	○	○	○	○	○

【SAMPLE】ブラウザの情報を調べる ・・・・・・・ P.216

NAVIGATOR.03

Javaが有効かどうかを調べたい

navigator.javaEnabled()

形式 メソッド

ブラウザがJavaをサポートし、使用可能か判断するプロパティです。使用可能な場合はtrue、使用不可の場合はfalseが返されます。Javaアプレットが使用できるかどうかによってページを移動させたい場合などに使用します。

文例

```
if(navigator.javaEnabled() == false) {
    alert("Javaが使用できません");
    history.back();
}
```

Javaアプレットが使用できない[illegible]は、「Javaが使用できません」というダイアログを表示し、1つ前のページに戻ります。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	[illegible]	[illegible]	[illegible]
	○	○	○	○	○	○	○	○	○

参照
MIME タイプの情報を調べたい・・・・・・・・・・・・P.214
[SAMPLE] ブラウザのプラグイン[illegible]べる・・・P.218

NAVIGATOR.04

プラグインの情報を調べたい

★.length　　プラグイン数を参照
◆.name　　プラグインの名前を参照
◆.filename　　プラグインのファイル名を参照
◆.description　　プラグインの詳細情報を参照

★……navigator.plugins
◆……navigator.plugins[参照番号]

形式　プロパティ

ブラウザにインストールされているプラグインに関する情報を参照するプロパティです。

文例

```
j = navigator.plugins.length;
```
プラグイン数を変数j に代入します。

```
alert("プラグイン名：" + navigator.plugins[n].name);
```
n+1 番目のプラグインの名前をダイアログに表示します。

```
document.write(navigator.plugins[i].filename);
```
i+1 番目のプラグインのファイル名を表示します。

```
document.write(navigator.plugins[i].description);
```
i+1 番目のプラグインのファイルの詳細情報を表示します。

Column　navigator.plugins

pluginsはPluginオブジェクト(p.065)の配列であり、配列の要素はブラウザにインストールされているプラグインに対応します。個々のオブジェクトはplugin[参照番号]で参照します。なお、最初のプラグインの参照番号は0です。

▶ブラウザ対応	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	×	×	×	○	○	○	△	○	×

※ Opera は description に非対応

MIME タイプの情報を調べたい ………… P.214
【SAMPLE】ブラウザのプラグイン情報を調べる ‥ P.218

NAVIGATOR.05

MIMEタイプの情報を調べたい

★.length　MIMEタイプの数を参照
◆.description　MIMEタイプの■■情報を参照
◆.type　MIMEタイプを参照
◆.enabledPlugin　MIMEタイプに対応するプラグインの使用の可否を参照
◆.suffixes　MIMEタイプの拡張子を■■

★……navigator.mimeTypes
◆……navigator.mimeTypes[参照番号]

形式 プロパティ

ブラウザがサポートするMIMEタイプ（インターネット上でデータの形式を指定する仕組み）に関する情報を参照するプロパティです。

descriptionプロパティ

MIMEタイプの詳細情報を参照します。

typeプロパティ

MIMEタイプ（image/gif、text/htmlなど）を参照します。

enablePluginプロパティ

MIMEタイプに対応するプラグインが使用可能か判断するプロパティです。指定したプラグインが使用できる場合はtrue、使用できない場合はfalseを返します。

suffixesプロパティ

MIMEタイプの拡張子を参照します。

文例

```
j = navigator.mimeTypes.length;
```

MIMEタイプ数を変数jに代入します。

```
document.write(navigator.mimeTypes[0].type);
```

1番目のMIMEタイプを表示します。

```
document.write(navigator.mimeTypes[i].description);
```

i+1番目のMIMEタイプの詳細情報を表示します。

```
if(navigator.mimeTypes["application/x-shockwave-flash"].enabledPlugin ==
true){
    alert("作品の感想をお待ちしています");
}
```

Flash プラグインが使用できる場合は「作品の感想をお待ちしています」というダイアログを表示します。

Column navigator.mimeTypes

mimeTypesはMimeTypeオブジェクトの配列であり、配列の要素はそのブラウザがサポートするMIMEタイプに対応します。個々のMIMEタイプはmimeTypes[参照番号]で参照します。なお、最初の要素の参照番号は0です。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	×	×	×	○	○	○	○	○	×

参照

プラグインの情報を調べたい・・・・・・・・・・・・・P.213
【SAMPLE】ブラウザのプラグイン情報を調べる・・P.218

ブラウザの情報を調べる

SAMPLE SCRIPT

ブラウザの情報を参照して表示するサンプルです。なお、使用言語についてはブラウザの対応が異なる（browserLanguageプロパティはFirefoxやChrome、languageプロパティはInternet Explorerに対応していない）ため、ブラウザを判別して使用するプロパティを分けています。

JavaScript

```
//ブラウザの情報を参照する関数
function pageLoad(){
    var html = ""; //htmlを追加していく変数
    var myAppName = navigator.appName; //ブラウザ名
    html += "<b>アプリケーション名(appName)：</b>" + myAppName + "<br />";
    html += "<b>バージョン(appVersion)：</b>" + navigator.appVersion + "<br />";
    html += "<b>コード名(appCodeName)：</b>" + navigator.appCodeName + "<br
/>";
    //FireFoxはbrowserLanguageプロパティは非対応
    if(myAppName.indexOf("Netscape") < 0){
        html += "<b>使用言語(browserLanguage)：</b>" + navigator.
browserLanguage + "<br />";
    }
    //Internet ExplorerはLanguageプロパティは非対応
    if(myAppName.indexOf("Microsoft") < 0){
        html += "<b>使用言語(language)：</b>" + navigator.language + "<br />";
    }
    html += "<b>プラットフォーム(platform)：</b>" + navigator.platform + "<br
/>";
    html += "<b>エージェント名(userAgent)：</b>" + navigator.userAgent + "<br
/>";
    document.getElementById("body").innerHTML = html;
                                    //作成したHTMLを<body>タグ内に挿入
}
```

HTML

```
<body id="body" onload="pageLoad()">
</body>
```

Internet Explorer
ブラウザの情報を調べる
アプリケーション名(appName):Microsoft Internet Explorer
バージョン(appVersion):5.0 (compatible; MSIE 10.0; Windows NT 6.2; WOW64; Trident/6.0; .NET4.0E; .NET4.0C)
コード名(appCodeName):Mozilla
使用言語(browserLanguage):ja-JP
プラットフォーム(platform):Win32
エージェント名(userAgent):Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; WOW64; Trident/6.0; .NET4.0E; .NET4.0C)

Firefox
ブラウザの情報を調べる
www.shoeisha.com/book/pc/dic/work/sample
アプリケーション名(appName):Netscape
バージョン(appVersion):5.0 (Windows)
コード名(appCodeName):Mozilla
使用言語(language):ja
プラットフォーム(platform):Win32
エージェント名(userAgent):Mozilla/5.0 (Windows NT 6.2; WOW64; rv:20.0) Gecko/20100101 Firefox/20.0

iphone
1:47
ブラウザの情報を調べる
www.shoeisha.com/
アプリケーション名(appName):Netscape
バージョン(appVersion):5.0 (iPhone; CPU iPhone OS 6_1_3 like Mac OS X) AppleWebKit/536.26 (KHTML, like Gecko) Version/6.0 Mobile/10B329 Safari/8536.25
コード名(appCodeName):Mozilla
使用言語(language):ja-jp
プラットフォーム(platform):iPhone
エージェント名(userAgent):Mozilla/5.0 (iPhone; CPU iPhone OS 6_1_3 like Mac OS X) AppleWebKit/536.26 (KHTML, like Gecko) Version/6.0 Mobile/10B329 Safari/8536.25

参照

NAVIGATOR.SAMPLE-02

ブラウザの プラグイン情報を調べる

ブラウザのプラグイン情報を書き出すサンプルです。Javaの有効／無効、プラグインの情報を表示します。プラグインはlengthプロパティを使いすべてのプラグインを参照しています。なお、プラグイン関係のプロパティはInternet Explorerには対応していないので情報を表示できません。

JavaScript

```
//ブラウザの情報を参照する関数
function pageLoad(){
    //変数の宣言
    var html = "";
    var boolJava;
    var dataTaint;
    //Javaが有効かどうか
    if(navigator.javaEnabled()){
        boolJava = "有効";
    }else{
        boolJava = "無効";
    }
    html += "<b>Java : </b>" + boolJava +" <hr />";
    html += "<b>Plugin</b><br />";
    html += "<table border='1'>"
    html += "<tr><th>種類<br/>mimeTypes.type</th>";
    html += "<th>詳細<br/>mimeTypes.description</th>";
    html += "<th>拡張子<br/>mimeTypes.suffixes</th>";
    html += "<th>ファイル名<br/>plugins.filename</th></tr>";
    //プラグインをすべて参照する
    for( i=0; i<navigator.plugins.length; i++ ){
        html += "<tr><td>" + navigator.mimeTypes[i].type + "</td>";
        html += "<td>" + navigator.mimeTypes[i].description + "</td>";
        html += "<td>" + navigator.mimeTypes[i].suffixes + "</td>";
        html += "<td>" + navigator.plugins[i].filename + "</td></tr>";
    }
    html += "</table>";
    document.getElementById("body").innerHTML = html;
                                        //作成したHTMLを<body>タグ内に挿入
}
```

HTML

```
<body id="body" onload="pageLoad()">
</body>
```

Internet Explorer

Java：有効

Plugin

種類 mimeTypes.type	詳細 mimeTypes.description	拡張子 mimeTypes.suffixes	ファイル名 plugins.filename

プラグイン関係のプロパティはInternet Explorerには対応していないので情報が表示されません

Firefox

Java：無効

Plugin

種類 mimeTypes.type	詳細 mimeTypes.description	拡張子 mimeTypes.suffixes	ファイル名 plugins.filename
application/x-vnd.google.update3webcontrol.3			npGoogleUpdate3.dll
application/x-vnd.google.oneclickctrl.9			NPSPWRAP.DLL

Firefoxではプラグイン情報が表示されます

参照
javaEnabled メソッド ………… P.212
length プロパティ ………… P.213
filename プロパティ ………… P.213
description プロパティ ………… P.213
type プロパティ ………… P.214
suffixes プロパティ ………… P.214

画像を扱いたい

★ = new Image()	画像オブジェクトを作成
◆.画像名	画像オブジェクトを作成
◆.images["参照番号"]	画像を参照
◆.images.length	画像の数を参照・設定

★……Imageオブジェクトを[illegible]する変数
◆……ドキュメントオブジェクト【省略可】

形式 オブジェクト(Image)、プロパティ(images.length)

新しい画像のオブジェクトを作成するには、newステートメント(p.039)を使ってImageオブジェクトを作成します。Imageは画像を扱うオブジェクトです。作成したオブジェクトはsrcプロパティ(p.223)を指定し、画像として利用できるようにします。

既存の画像を参照するには、画像の名前や参照番号を利用します。画像名は<img>タグのname属性あるいはid属性で指定されている名前です。たとえば、と指定されている[illegible]を参照する場合は次のどちらかのようになります。

```
document.profImg
document.images["profImg"]
```

参照番号を利用する方法は、ドキュメント中の[illegible]を要素とする配列から、その参照番号に対応する画像にアクセスする仕組みです。images[参照番号]という形式で指定します。画像の参照番号はドキュメントの中に画像が記述されている順番で、0からの連番になります。たとえば、ドキュメント中の2番目の画像を参照する場合は次のようになります。

```
document.images[1]
```

文例

```
myImage = new Image();
```

Imageオブジェクトmylmageを作成します。

```
document.images[1].height = 250;
```

2番目の画像の高さを250ピクセルに設定します。

```
alert(document.images.length);
```

ドキュメント中の画像の総数をダイアログに表示します。

Column　画像をDOMで参照する

と指定されている画像をDOMで参照する場合は次のようになります。

```
document.getElementById("profImg")
```

ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照

【SAMPLE】画像の情報を表示する ･････････ P.225

IMAGE.02

画像の情報を扱いたい

★.name 画像の名前を参照
★.width 画像の幅を参照／設定
★.height 画像の高さを参照／設定

★……Imageオブジェクト（画像名またはimages["参照番号"]）

形式 プロパティ

画像のサイズや枠線の幅、周りの文章などとの間隔に関する情報を参照／設定するプロパティです。たとえば画像のサイズに合わせてウィンドウのサイズを変更する場合などに利用します。

文例

```
myImg = document.images[0].name;
```
1番目の画像の名前を変数myImgに代入します。

```
document.write("画像の幅は" + document.images[i].width + "、高さは" +
document.images[i].height + "です");
```
i+1番目の画像の幅と高さを書き出します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

参照
画像のURLを参照／設定したい…………P.223
【SAMPLE】画像の情報を表示する………P.225

IMAGE.03

画像のURIを参照／設定したい

★.src　画像のURIを参照／設定

★.lowsrc　低解像度用画像のURIを参照／設定

★……Imageオブジェクト（画像名またはimages["参照番号"]）

形式　プロパティ

Imageオブジェクトの URIを参照／設定するプロパティです。プロパティの値は絶対URIまたは相対URIで指定します。

srcプロパティ

画像のURIを参照／設定します。状況に応じて絶対URIか相対URIを用います。srcプロパティを変更すると、表示後に画像を切り替え可能になり、アニメーションを作成できます。

lowsrcプロパティ

低解像度用の画像を設定します。

文例

```
new Image().src = "product01.jpg";
```

Imageオブジェクトを作成し、URI をproduct01.jpgに設定することで、その画像を表示します。

```
myImg = document.images["product05"].src;
```

変数myImgに画像product05のURIを代入します。

ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照　画像の情報を扱いたい……P.222

画像の読み込みの完了を調べたい

★.complete　画像の読み込みの完了を参照

★……Imageオブジェクト(画像名またはimages["参照番号"])

形式 プロパティ

画像の読み込みが完了している場合は値としてtrue、完了していない場合はfalseを返します。画像を使ったゲームやアニメーションなどで、すべての画像の読み込みが完了してから処理を開始したい場合に利用します。

文例

```
if(document.fakeImg.complete) alert("読み込み完了！");
```

画像fakeImgの読み込みが完了したら、ダイアログを表示します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照

画像が読み込めないときに処理を行いたい・・・P.130
読み込み時や移動時に処理を行いたい・・・・・・P.128

IMAGE.SAMPLE-01

画像の情報を表示する

画像の大きさを表に書き出すサンプルです。for文でページ上のすべての画像の名前とサイズを取得し、innerHTMLで<div id="div1"></div>のタグ内に出力しています。

JavaScript

```
function pageLoad() { //画像情報を取得してdivタグに表示する関数
    for (var rows = "", i = 0; i < document.images.length; i++) {
        var img = document.images[i]; // 画像の参照を取得
        rows += "<tr><td>" + img.alt + "</td>";
        rows += "<td>" + img.width + "×" + img.height + "</td></tr>";
    }
    document.getElementById("div1").innerHTML = "<table>" + rows + "</
table>";
}
```

HTML

```
<body id="body" onload="pageLoad()">
<div>
    <img src="image/usagi.gif" id="usagi"  alt="usagi" />
    <img src="image/neko.gif" id="neko"  alt="neko" />
    <img src="image/panneko.gif" id="panneko"  alt="panneko" />
</div>
<hr />
<div id="div1"></div>
</body>
```

Internet Explorer

length プロパティ ・・・・・・・・・・・・・・・・・・ P.220
width プロパティ ・・・・・・・・・・・・・・・・・・ P.222
height プロパティ ・・・・・・・・・・・・・・・・・・ P.222

LINK.01

URIを参照／設定したい

location = ★ URIを参照／設定
URL = ★ URIを参照

★……オブジェクト名(Document/Window/Frameオブジェクト)

形式 プロパティ

URIを参照または設定するプロパティです。ボタンにリンクを設定したり、イベント発生時にページを移動させるなど、<a>タグ以外でページを移動させたい場合に利用できます。なお、locationプロパティの参照とURIプロパティは同じ働きになります。

locationプロパティ

URIを参照／設定します。locationプロパティでURIを設定すると、現在のページから指定したURIのページに移動できます。なお、ブラウザによってはlocationにURIを指定しても[illegible]作しないことがあります。その場合はlocation.hrefでURIを指定してください(p.228)。

URLプロパティ

ドキュメントのURIを返します。参照のみで設定はできません。

文例

```
<input type="button" value="ジャンプ" onclick="window.location.href='http://www.ank.co.jp/'" />
```

クリックされたら、指定のURI に移動します。

```
alert(document.URL);
```

現在のURI をダイアログに表示します。

ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

参照

ドメイン名を参照したい・・・・・・・・・・・・・・・・P.060
ページをリロードしたい・・・・・・・・・・・・・・・・P.227
ページ中のリンク[illegible]報を参照したい・・・・・・・・P.228
ページのURI を変更したい・・・・・・・・・・・・・・P.232
【SAMPLE】URI を参照／設定する・・・・・・・・・P.233

LINK.02

ページをリロードしたい

location.reload()

形式 オブジェクト

Locationは現在のページのURIに関する情報を扱うオブジェクトです。Locationオブジェクトのreloadメソッドでは、ページのリロード(再読み込み)が可能です。一定時間ごとに情報が書き換えられるページ(チャットなど)でタイマーと組み合わせて使用すれば、自動的に最新の情報を表示できます。

文例

```
<input type="button" value="リロード" onclick="location.reload()" />
```

ボタンがクリックされたら現在のページをリロードします。

ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照

URI を参照/設定したい ・・・・・・・・・・・・・・・ P.226

LINK.03

ページ中のリンク情報を参照したい

document.links[参照番号].★ ◆.href

リンクの情報を参照

リンクを参照／設定

★……プロパティ、メソッド
◆……locationまたはdocument.links[参照番号]

形式 プロパティ

Linksオブジェクト

Linkはドキュメント中にあるリンクを表すオブジェクト(<a href="">で定義されたリンク)です。linksプロパティはLinkオブジェクトの配列で、ドキュメント内のすべてのリンク情報を持っています。各リンクの情報は配列の要素として格納されていて、document.links[参照番号] という形式で参照できます。リンクの参照番号はドキュメントの中でリンクが記述されている順番で、0からの連番になります。

hrefプロパティ

href属性を表すプロパティで、リンク先または現在のページのURIを参照／設定します。

文例

```
document.write(document.links.length);
```
ドキュメント内のリンク数を書き出します。

```
newWin.location.href = "http://www.ank.co.jp/";
```
ウィンドウnewWinのURIを設定します。

Column　Areaオブジェクト

イメージマップ(クリッカブルマップ)の<area>タグで定義されるエリア(リンク領域)の情報を持っているオブジェクトに、Areaオブジェクトがあります。Areaオブジェクトのリンクはlinkオブジェクトに内包されるため、実際に利用する場合はエリア名またはリンク配列であるlinks[参照番号]の形式で指定します。なお、Areaオブジェクトの持つプロパティはlengthプロパティをのぞきLinkオブジェクトと共通です。

▶ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照

ページ中のアンカー情報を参照したい ······· P.230
URI を参照／設定したい ··················· P.226
【SAMPLE】ページのリンクを書き出す ······ P.235

LINK.04

リンクの読み込み先を設定したい

document.links[参照番号].target　読み込み先を参照／設定

形式 プロパティ

リンク先のウィンドウ名やフレーム名を参照／設定します。

文例

```
document.links[0].target = "subWin";
```

1番目のリンクの読み込み先をウィンドウ名subWinにします。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照 URIを参照／設定したい …………… P.226

LINK.05

ページ中のアンカー情報を参照したい

document.anchors[参照番号].★

★……プロパティ

形式 プロパティ

Anchorはドキュメント中にあるアンカーを表すオブジェクト(<a name="">で定義されたアンカー)です。anchorsプロパティはAnchorオブジェクトの配列で、ドキュメント内におけるすべてのアンカーの情報を持っています。各アンカーの情報は配列の要素として格納されていて、document.anchors[参照番号] という形式で参照できます。アンカーの参照番号はドキュメントの中でアンカーが記述されている順番で、0からの連番になります。

文例

```
document.anchors[1].href = "070831";
```

2番目のリンク先のアンカー名を070831にします。

```
len = document.anchors.length;
```

ドキュメント内のアンカー数を変数lenに代入します。

▶ ブラウザ対応表	IE10	IE9	[illegible]	[illegible]	Chrome	Safari	[illegible]	[illegible]	Android
	○	○	○	○	○	○	○	○	○

ページ中のリンク情報を参照したい ········ P.228

LINK.06

ページのロケーション情報を参照したい

★.pathname	パス名を参照
★.host	ホストを参照
★.hostname	ホスト名を参照
★.port	ポート番号を参照
★.protocol	プロトコルを参照

★……locationまたはdocument.links[参照番号]

形式 プロパティ

ページのロケーションに関する情報を参照します。現在のページのロケーションの場合は★にlocationを、ページ上のリンクオブジェクトを参照する場合は★に document.links[参照番号] を指定します。ページ中にある最初のリンクの参照番号は0になります。

文例

```
subWin.document.write("ホスト名：", location.host);
```

ホスト名をウィンドウ名subWinに表示します。

```
subWin.document.write("プロトコル：", document.links [i].protocol);
```

i+1番目のリンクのプロトコルをウィンドウ名subWinに表示します。

対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照
ドメイン名を参照したい ･････････････････ P.060
ページ中のリンク情報を参照したい ･････････ P.228
【SAMPLE】ページのリンクを書き出す ･･････ P.235

LINK.07

ページのURIを変更したい

location.replace(★)

★……移動先のURI

形式 メソッド

指定したページ★へ移動するメソッドです。replaceメソッドでページを変更すると、Historyオブジェクトを上書きするため、履歴に元のページが残りません。つまり、ブラウザの[戻る]ボタンを使って現在のページに戻れなくなります。

文例

```
<input type="button" value="リロード" onclick="location.replace('http://www.ank.co.jp/')" />
```

ボタンがクリックされたら指定のページへ移動します。

ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照
URI を参照／設定したい ……………… P.226
【SAMPLE】履歴を残さずページを移動する … P.237

LINK.SAMPLE-01

URIを参照／設定する

ボタンや画像のクリック、またはセレクトボックスの操作でリンクを実行するサンプルです。onclick属性やonchange属性に設定した関数内でlocationプロパティを変更することによって、<a>要素以外の要素にもリンクを設定できます。

JavaScript

```
function go(index) { // 指定のページをiframeに表示する
    var win = document.getElementById("myFrame").contentWindow;
    switch (index) {
        case 0:
            break;
        case 1:
            win.location = "http://www.ank.co.jp/";
            break;
        case 2:
            win.location = "http://www.shoeisha.co.jp/";
            break;
    }
}
function selectURI() { // セレクトボックスで選択された項目のページを表示
    go(document.getElementById("select").selectedIndex);
}
```

HTML

```
<body>
<form>
    <input type="button" value="アンク" onclick="go(1);" /><br>
    <input type="button" value="翔泳社" onclick="go(2);" /><br>
    <select id="select" onchange="selectURI();">
            <option selected="selected">選択して下さい</option>
            <option>アンク</option>
        <option>翔泳社</option>
    </select>
    <p><img src="images/anklogo.gif" alt="ANKロゴ"  onclick="go(1);" /></p>
    <iframe id="myFrame"></iframe>
</form>
</body>
```

ボタンや画像をクリック、もしくはセレクトボックスを選択すると、iframeにリンクが表示されます

location オブジェクト ················ P.226

LINK.SAMPLE-02

ページのリンクを書き出す

ドキュメント中のリンク情報を別ウィンドウに書き出すサンプルです。[リンク情報一覧]ボタンがクリックされた際にshowInfo関数を呼び出し、新しいウィンドウにページ上のリンクの情報を書き出します。

JavaScript

```
//ロケーション情報を取得する関数
function showInfo(){
	var infoWin = window.open("", "infoWin", "width=320,height=500");
	infoWin.document.write("「", document.title, "」のリンク一覧<hr />");
	urls = new Array(document.links.length);
	//ページ上のすべてのリンクのhref属性の値を取得
	for( i=0; i<document.links.length; i++ ){
		urls[i] = document.links[i].href;
	}
	//新しく開いたウィンドウにロケーション情報を表示
	for( i=0; i<document.links.length; i++ ){
		infoWin.document.write("<p>リンク No.", i +1, "<br />");
		infoWin.document.write("リンク先=<a href='", urls[i], "' target='_blank'
>");
		infoWin.document.write(urls[i], "</a><br />");
		infoWin.document.write("ホスト名=", document.links[i].hostname,
"<br />");
		infoWin.document.write("プロトコル=", document.links[i].protocol, "</
p>");
	}
	infoWin.document.bgColor = "#ffcc00";
}
```

HTML

```
<body>
    <p><b>ページ中のロケーション情報・リンク一覧を別ウィンドウに表示します</b></p>
    <form action="">
        <p><input type="button" value="リンク情報一覧" onclick="showInfo()"
/></p>
    </form>
    <ul>
        <li><a href="http://www.google.co.jp/">Google</a></li>
        <li><a href="http://www.yahoo.co.jp/">Yahoo! Japan</a></li>
        <li><a href="http://www.infoseek.co.jp/">infoseek</a></li>
        <li><a href="http://www.goo.ne.jp/">goo</a></li>
    </ul>
</body>
```

Internet Explorer

[リンク情報一覧]ボタンをクリックすると、
新しいウィンドウにリンク先の情報が表示されます

参照
links オブジェクト …… P.228
href プロパティ …… P.228
hostname プロパティ …… P.231
protocol プロパティ …… P.231

LINK.SAMPLE-03

履歴を残さずページを移動する

履歴を残さずにページを移動するサンプルです。body要素のonload属性にタイマーをセットして、3秒後にアンクのWebサイトへ移動させています。replaceメソッドでは履歴が残らないため、ブラウザの[戻る]ボタンでこのサンプルのページへ戻ることはできません。

HTML

```
<body onload='setTimeout(function() {
        location.replace("http://www.ank.co.jp/");
        }, 3000)'><!--タイマーをセット-->
        <p><b>3秒後にページを移動します</b></p>
</body>
```

ページをロードした3秒後に指定したWebサイトに移動します

移動した後は[戻る]ボタンをクリックしてもサンプルのページに戻りません

replace メソッド ・・・・・・・・・・・・・・・・・・・・ P.232

HISTORY.01

どのページから来たのか調べたい

★.referrer　　リンク元のURIを参照

★……Documentオブジェクト(ドキュメント名)

形式　プロパティ

ハイパーリンクによる移動元のページを調べます。

Webブラウザのアドレスバーで直接URIを入力して現在のページを開いたり、[戻る]ボタンを使用したりした場合はリンク元(前のページ)のURIは取得できません。また、ローカルファイルについてもURIは取得できません。

文例

```
document.write(document.referrer + "からいらっしゃいましたね！");
```

リンク元のURIを書き出します。

```
ref = document.referrer;
```

リンク元のURI を変数ref に代入します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

参照

URI を参照／設定したい ・・・・・・・・・・・・・・・・ P.226
【SAMPLE】どのページから来たのか調べる ・・・ P.242

HISTORY.02

履歴の数を調べたい

★.history.length

★……Windowオブジェクト（ウィンドウ名）【省略可】

形式 プロパティ

lengthプロパティを利用すれば、ブラウザがその時点で持っている履歴の総数を参照できます。なお、履歴が存在しない場合のlengthプロパティの値はブラウザによって異なるので注意が必要です。Internet Explorer とOperaでは0であるのに対し、Firefoxでは1になります。

文例

```
len = history.length
```

履歴の総数を変数len に代入します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照
ブラウザのボタンと同様の処理をしたい …… P.083
履歴の前後に移動したい …… P.240
【SAMPLE】履歴の前後に移動する …… P.244

HISTORY.03

履歴の前後に移動したい

★.history.back()	1つ前のページに戻る
★.history.forward()	1つ後のページに進む
★.history.go(▲)	指定した数だけ移動

★……Windowオブジェクト(ウィンドウ名)【省略可】
◆……移動するページ数(正または負の整数)

形式 メソッド

Historyオブジェクトはブラウザの履歴に関する情報を持ったオブジェクトです。履歴とはブラウザの[進む]ボタンや[戻る]ボタンのクリックで表示される前後に表示したページの記録です。Historyオブジェクトを使用すると、前後に表示したページに移動できます。

backメソッド

1つ前のページに戻るメソッドです。ブラウザの[戻る]ボタンのクリックと同様の効果を実現します。

forwardメソッド

1つ後のページに進むメソッドです。ブラウザの[進む]ボタンのクリックと同様の効果を実現します。

goメソッド

引数に指定した数だけページを移動します。負の値を指定すると、指定した数だけ前のページに戻ります。ただし、履歴の数より大きい数値を指定した場合やブラウザを起動して最初にページを開いたときなど、指定した履歴が存在しない場合にはメソッドを実行しても何も起こらないので、lengthプロパティであらかじめ履歴の数を確認してから使用するとよいでしょう(p.263)。

文例

```
<input type="button" value="戻る" onclick="history.back()" />
```

［戻る］ボタンがクリックされると、１つ前のページに戻ります。

```
<input type="button" value="進む" onclick="history.forward()" />
```

［進む］ボタンがクリックされると、１つ後のページに進みます

```
history.go(2);
```

2つ後のページに進みます。

```
history.go(-2);
```

2つ前のページに戻ります。

```
history.go(jumpNum);
```

変数jumpNum の値で指定された数だけ履歴を移動します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照

ブラウザのボタンと同様の処理をしたい・・・・・P.083
履歴の数を調べたい・・・・・・・・・・・・・・・・・・・・P.239
【SAMPLE】履歴の前後に移動する・・・・・・・・・P.244

HISTORY.SAMPLE-01

どのページから来たのか調べる

リンク元のページのURIを書き出すサンプルです。リンク元のページにあるリンクではHTML/XHTMLを利用し、a要素のhref属性でリンク先のURIを設定しています。一方、inputボタンでは、locationプロパティの値にリンク先のURIを設定することで移動できるようにしています。移動先のページでは、referrerプロパティでリンク元のページのURIを書き出します。

referrerプロパティはWebサーバ上でのみ参照できます。ただし、Webサーバの設定により参照できない場合があります。

なお、別のページからのリンクではなくURIの直接入力やブラウザの「戻る」ボタン、locationプロパティの変更などによって移動した場合は、元ページのURIを参照できません。

JavaScript

※ リンク先のhistory_reffer.htmlで呼び出される

```
var uri;
//参照できない場合
if(document.referrer.length == 0){
    uri = "参照できません";
//参照できた場合
}else{
    uri = document.referrer;
}
//リンク元のURIを取得して表示
document.write("リンク元のページ：", uri);
```

HTML

※ リンク元のHTML

```
<body>
    <p><a href="history_referrer.html">移動(移動先をa要素のhref属性で設定)</a></
p>
    <p><input type="button" value="移動(移動先をlocationプロパティに代入)"
onclick="location = 'history_referrer.html'" /></p>
</body>
```

Internet Explorer

リンク元のページ

リンク先のページ。a要素をクリックしてリンクすると、referrerプロパティによってリンク元のURIを表示します

リンク先のページ。input要素（下のボタン）をクリックするとlocationプロパティによってリンクするので、referrerプロパティはリンク元のURIを参照できません

参照 referrer プロパティ ・・・・・・・・・・・・・・・・・・ P.238

HISTORY.SAMPLE-02

履歴の前後に移動する

history.backメソッドを使って前のページに戻るサンプルです。履歴がある場合は、[進む]、または[戻る]ボタンでページ移動が可能です。また、history.lengthプロパティで履歴数を参照し表示しています。履歴がない場合、Internet ExplorerとOperaでは値が0になりますが、Firefoxでは1になります。

[新しいページへ]ボタンがクリックされると、JavaScriptで作成されたページへ移動します。そのページの[戻る]ボタンがクリックされるとhistory.back関数が呼び出され、最初のページへ戻ります。

JavaScript

```
//履歴数を参照して表示する関数
function loadPage(){
    var x = history.length;
    var html = "<table border=1>";
    html += "<tr><td>履歴数</td><td>" + x + "</td></tr></table>";
    //作成したHTMLを挿入
    document.getElementById("p2").innerHTML = html;
}

//新しいページへ遷移させる関数
function clickNewPage(){
    document.open();
    document.write("JavaScriptで作成された新しいページです。「戻る」ボタンで戻ります。
<br />");
    document.write("<input type='button' value='戻る' onclick='history.back()' />");
    document.close();
}
```

HTML

```
<body onload="loadPage()">
    <p>
    <input type="button" value="新しいページへ" onclick="clickNewPage()" />
    <input type="button" id="forward" value="進む" onclick="history.
forward()" />
    <input type="button" id="back" value="戻る" onclick="history.back()" />
    </p>
    <hr />
    <p id="p2">
    </p>
</body>
```

Internet Explorer

新しいページへ　進む　戻る

履歴数 4

[新しいページへ]をクリックすると、JavaScriptで作成したページへ移動します

JavaScriptで作成された新しいページです。「戻る」ボタンで戻ります。

戻る

[戻る]ボタンをクリックすると、元のページに戻ります

back メソッド・・・・・・・・・・・・・・・・・・・・・P.240
forward メソッド・・・・・・・・・・・・・・・・・・・P.240
length プロパティ・・・・・・・・・・・・・・・・・・P.239

CHANGE.01

リンクで何も動作させたくない

void(★) 値を返さない

★……メソッドや数式など

形

void関数は何も値を返さない命令です。引数★に指定した関数やメソッド、数式は実行されますが、それによって値は返されません。void(0)と指定すると、0という評価はされますがスクリプトにまったく影響を与えない命令になります。リンクがクリックされたときに何も動作させたくない場合などに使用します。

文例

```
<a href="javascript:void(0)">クリックしても何もおこりません。</a>
```

リンクがクリックされても何も動作しないようにします。

Column リンクのJavaScript

上記の文例のように<a>タグのhref属性で指定するリンク先にjavascript:[JavaScriptのコード]と記述すると、指定したJavaScriptの処理を実行できます。たとえば

```
<a href="javascript:location.reload();">再読み込み</a>
```

のように記述すると、リンクがクリックされたときにページをリロード(location.reload())できます。

しかし、このような方法はa要素本来の意図と異なり、JavaScriptが無効の場合にはデッドリンク(リンク切れ)にもなるので、あまり好ましい利用法ではありません。使用にあたっては十分注意してください。

▶ ブラウザ対応表	IE10	[illegible]	IE8	Fx	[illegible]	[illegible]	Opera	[illegible]	Android
	○	○	○	○	○	○	○	○	○

(基本編) JavaScript 記述の注意点／予約語 ‥ P.009

CHANGE.02

文字列を数値に変換したい

● = parseFloat(★)　浮動小数点数に変換
● = parseInt(★, ◆)　整数に変換

●……変換された数値
★……数値を表す文字列
◆……基数【省略可】

形式　関数

文字列を浮動小数点数(小数)や整数に変換するメソッドです。

parseFloat関数

文字列を浮動小数点数に変換します。

parseInt関数

文字列を整数に変換します。基数◆を指定すると、◆進数で表記された文字列を10進法の整数に変換できます(小数点以下は切り捨て)。基数を省略すると文字列は10進数とみなされます。

たとえば、以下の例ではどれも変数iには255が代入されます。

i = parseInt("ff ",16)	→16進数ff を10進法に変換して代入
i = parseInt("255.11",10)	→小数点以下を切り捨てて整数に変換して代入
i = parseInt("377",8)	→8進数377を10進法に変換して代入
i = parseInt("11111111",2)	→2進数11111111を10進法に変換して代入

文例

```
myVar = parseFloat("123.45");
```

文字列123.45 を浮動小数点数に変換して、変数myVar に代入します。

```
myVar = parseInt("1111", 2);
```

文字列1111 を2 進数の値として10 進法に変換し、変数myVar に代入します。

ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照
数値を文字列に変換したい ・・・・・・・・・・・・・・ P.248
数式を数値に変換したい ・・・・・・・・・・・・・・・ P.249
数値かどうかを調べたい ・・・・・・・・・・・・・・・ P.251
【SAMPLE】文字列を数値に変換する ・・・・・・・ P.253

CHANGE 04

数式を数値に変換したい

eval(★)

★……メソッドや数式など

書式 関数

数式や文字列をJavaScriptの構文として実行します。フォームに入力された数式を演算したい場合などに利用します。

たとえば、以下の例では1行目は「1+1」と表示され、2行目は「2」となります。

```
i = "1+1";
document.write(i);
document.write(eval(i));
```

以下の例では、"15"を16進数表記したときの文字列"f"が変数jに代入されます。

```
j = eval("15").toString(16);
```

文例

```
m = eval("1*2+3");
```

1*2+3 を数式に変換し、演算結果を変数m に代入します。

```
document.write(eval("123").toString(16));
```

数値に変換された123 を16 進数表記の文字列に変換し、書き出します(結果は7b)。

▶ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照
文字列を数値に変換したい ……………… P.247
数値を文字列に変換したい ……………… P.248
数値かどうかを調べたい ………………… P.251
【SAMPLE】文字列を数値に変換する ……… P.253

CHANGE.05

文字列をエンコード／デコードしたい

● = escape(★)　文字列をエンコード
● = unescape(★)　文字列をデコード

●……エンコード／デコードされた文字列
★……文字列

形式 関数

文字列をエンコード／デコードする関数です。これらの関数は主にクッキーやフォーム内容の送信の際、文字列中の特殊文字や漢字をアルファベットなどのエスケープ文字(文字コード)に変換するために使われます。たとえば、クッキーなどにアルファベットと数値以外のデータを書き込む際にはescape関数を使います。クッキーから読み込んだデータを元の文字列に戻す際にはunescape関数を使います。

escape関数

文字列をエスケープ文字にエンコードします。数値とアルファベットはそのままですが、その他の文字は%xx(xxは16進数)のような形式に変換します。ただし、ブラウザにより文字コードが異なる場合があります。

unescape関数

エンコードされた文字列をデコードして、元の文字列に戻します。

文例

```
myStr = "あつい"
myVal = escape(myStr);
```

Stringオブジェクトmy Strの内容をエンコードして変数myValに代入します(値は%u3042%u3064%u3044)。

```
document.write(unescape("%u306D%u3080%u3044"));
```

文字コード"%u306D%u3080%u3044"を文字列に変換して書き出します(結果は「ねむい」)。

▶ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

参照
文字コードを扱いたい・・・・・・・・・・・・・・・・・・P.203
使用できる数値の範囲を調べたい・・・・・・・・・・P.266

CHANGE.06

数値かどうかを調べたい

● = isNaN(★)

●……結果(trueまたはfalse)
★……数値や文字列

形式 関数

指定した値★が数値ではない場合はtrue、数値の場合はfalseを返す関数です。なお、NaNはNot a Numberの略です。

文例

```
myVal = isNaN(n);
```

変数n の値が数値かどうかを調べ、結果を変数myVal に代入します。

```
if(isNaN(document.form1.txt1.value)){
    alert(数値を入力してください);
}
```

入力された値が数値ではない場合、「数値を入力してください」というダイアログを表示します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照

文字列を数値に変換したい …………… P.247
数値を文字列に変換したい …………… P.248
数式を数値に変換したい …………… P.249

CHANGE.07

真偽値を作成したい

★ = new Boolean(◆)

★……Booleanオブジェクトを格納する変数
◆……値[[illegible]可]

形式 オブジェクト

Boolean（真偽：ブーリアン）はtrueまたはfalseを値に持つオブジェクトです。新しいBooleanオブジェクトを作成するには、newステートメント(p.037)を使用しますが、明示的にBooleanオブジェクトを作成することはあまりありません。

Booleanオブジェクトは、状況に応じて異なる処理を行いたい場合に、ifステートメント(p.024)とともに使用されます。

値◆を省略するか0や空の文字列("")を指定した場合はfalse、それ以外の場合はtrueを返します。

文例

```
myBool = new Boolean();
```
Booleanオブジェクトを myBoolを作成します。

```
myBool = new Boolean(true);
```
Booleanオブジェクト myBoolをtrueにします。

▶ ブラウザ[illegible]	IE10	IE9	[illegible]	Fx	[illegible]	Safari	[illegible]	[illegible]	Android
	○	○	○	○	○	○	○	○	○

参照
（基礎編）オブジェクトを扱う …………… P.037

CHANGE.SAMPLE-01

文字列を数値に変換する

2つの数字の和、差、積、商の計算およびn進数への変換を行うサンプルです。入力欄に数値を入力し、[計算]ボタンをクリックするとcalc関数によって演算が実行されます。また、[n進数に変換/戻す]ボタンのクリックによって計算結果値をn進数に変換できます。

JavaScript

```
window.onload = loadPage;

//変数の宣言
var result="";

function loadPage(){
    formElem = document.getElementById("form1");
}

//入力された値を計算する関数
function calcNum(){
    //計算結果を変数resultに代入
    result = eval(formElem.num1.value +
            formElem.select1.options[formElem.select1.selectedIndex].value +
            formElem.num2.value);
    formElem.result1.value = result;
}

//計算結果の値をn進数に変換する関数
function change(num){
    calcNum();
    var fNum = parseFloat(result);
    formElem.result1.value = fNum.toString(num);
}
```

HTML

```
<body>
  <form action="" id="form1">
    <p>
    <input type="text" name="num1"/>
    <select name="select1">
      <option value="+">+</option>
      <option value="-">−</option>
      <option value="*">×</option>
      <option value="/">÷</option>
    </select>
    <input type="text" name="num2"/>
    <input type="button" value="計算" onclick="calcNum()" />
    </p>
    <hr />
    <p>
    結果：<input type="text" name="result1" onchange="this.value=result;"/>
    <input type="button" value="１０進数に戻す" onclick="change(10)" />
    </p>
    <p>
    <input type="button" value="２進数に変換" onclick="change(2)" />
    <input type="button" value="８進数に変換" onclick="change(8)" />
    <input type="button" value="１６進数に変換" onclick="change(16)" />
    </p>
  </form>
</body>
```

❶計算が実行されます

❷計算結果が16進法に変換されます

参照
parseFloat メソッド ・・・・・・・・・・・・・・・・ P.247
toString メソッド ・・・・・・・・・・・・・・・・・・ P.248
eval メソッド ・・・・・・・・・・・・・・・・・・・・・ P.249

MATH.01

乱数を発生させたい

● = Math.random()

●……乱数

形式 メソッド

randomメソッドは0から1の範囲(1は含まない)の乱数を発生させます。たとえば、一定の範囲のランダムな整数を得たい場合には、randomメソッドで返された値に範囲の上限の数を乗じ、floorメソッドやceilメソッド(次項を参照)を使って小数点以下を処理します。

例1：変数iに0から2までの範囲のランダムな整数(0,1,2)を代入します。

```
i = Math.floor(Math.random() * 3);
```

例2：変数i に3から5までの範囲のランダムな整数(3,4,5)を代入します。

```
i = Math.floor(Math.random() * 3) + 3;
```

文例

```
myMath = Math.random();
```

0から1未満の乱数を変数myMath に代入します。

```
myMath = Math.floor(Math.random() * 100) + 1;
```

1から100までの乱数を変数myMath に代入します。

Column　Mathオブジェクト

Mathは数学関数を扱うオブジェクトです。Mathオブジェクトには三角関数、平方根、累乗、対数などの演算を行うプロパティやメソッドがあります。新しいMathオブジェクトを作成する際はnewステートメント(p.037)を使用する必要はなく、直接プロパティやメソッドを記述して使用します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

参照

小数点以下を処理したい……………… P.257
【SAMPLE】ランダムに表示した 2 つの数字を比較する……………… P.267

MATH.02

小数点以下を処理したい

● = Math.ceil(★)　小数点以下を切り上げる
● = Math.floor(★)　小数点以下を切り捨てる
● = Math.round(★)　小数点以下を四捨五入する

●……結果として得られる値
★……数値または数式

形式 メソッド

計算結果の小数点以下を処理したり、randomメソッドで発生させた乱数から整数を取得したりする場合などに使用するメソッドです。★が文字列の場合、NaN(p.251)を返します。

ceilメソッド

小数点以下を切り上げ、数値を整数に変換します。

floorメソッド

小数点以下を切り捨て、数値を整数に変換します。

roundメソッド

小数点以下の四捨五入を行い、数値を整数に変換します。

文例

```
myMath = ceil(12.012);
```
12.012 の小数点以下を切り上げて整数に変換し、変数myMathに代入します(値は12)。

```
myMath = floor(num);
```
変数num の値の小数点以下を切り捨てて整数に変換し、変数myMathに代入します。

```
document.form1.round_out.value = Math.round(num);
```
変数num の値を四捨五入してフォームの出力欄(エレメント名round_out)に表示します。

▶ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照
乱数を発生させたい……P.256
使用できる数値の範囲を調べたい……P.266
【SAMPLE】ランダムに表示した２つの数字を比較する……P.267

MATH.03

絶対値を求めたい

● = Math.abs(★)

●……結果として得られる値
★……数値または数式

形式 メソッド

指定した数値の絶対値を求めるメソッドです。演算の結果などを必ず正の値で取得したい場合に使用します。★が文字列の場合、NaN(p.251)を返します。

文例

```
myMath = Math.abs(-78);
```
-78 の絶対値を変数myMath に代入します(値は78)。

```
myMath = Math.abs(x-y);
```
x-yの結果の絶対値を求め、変数myMath に代入します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

使用できる数値の範囲を調べたい ･････････ P.266

MATH.04

円周率を使いたい

Math.PI

形式 プロパティ

円周率(3.141592653589793)を返すプロパティです。返される値は小数点以下15桁です。

文例

```
pi = Math.PI;
```

円周率を変数piに格納します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

MATH.05

三角関数を使いたい

```
● = Math.sin(★)
● = Math.cos(★)
● = Math.tan(★)
● = Math.asin(★)
● = Math.acos(★)
● = Math.atan(★)
● = Math.atan2(★,◆)
```

式	説明
● = Math.sin(★)	正弦(サイン)を算出
● = Math.cos(★)	余弦(コサイン)を算出
● = Math.tan(★)	正接(タンジェント)を算出
● = Math.asin(★)	逆正弦(アーク・サイン)を算出
● = Math.acos(★)	逆余弦(アーク・コサイン)を算出
● = Math.atan(★)	逆正接(アーク・タンジェント)を算出
● = Math.atan2(★,◆)	逆正接(アーク・タンジェント)を算出

●……結果として得られる値
★……数値または数式
◆……数値または数式

形式 メソッド

引数に与えられた数値★を基に、三角関数および逆三角関数による演算を行うメソッドです。返される値については下記を参照してください。★が文字列の場合、NaN(p.251)を返します。

三角関数(sin、cos、tan)メソッド

引数は角度で、ラジアン単位で指定します。

逆三角関数(asin、acos、atan、atan2)メソッド

sinメソッドとcosメソッドの逆関数であるasinメソッドとacosメソッドでは、引数に-1から1の間の数値を指定する必要があります。結果はラジアン単位の角度が返されます。atan2メソッドは(★, ◆)座標の角度を返します。

メソッド	返される値
sin()	-1～1
cos()	-1～1
tan()	-∞～∞
asin()	-PI/2～PI/2
acos()	0～PI
atan()	-PI/2～PI/2
atan2()	-PI～PI

文例

```
document.write(Math.sin(1));
```
1のサインを求めて表示します。

```
document.write(Math.cos(num));
```
変数numの値のコサインを求めて表示します。

```
document.write(Math.tan(1));
```
1のタンジェントを求めて表示します。

```
b = Math.asin(1);
```
1のアーク・サインを求めて変数bに代入します。

```
myMath = Math.acos(num);
```
変数num のアーク・コサインを求めて変数myMathに代入します。

```
myMath = Math.atan(1);
```
1 のアーク・タンジェントを求めて変数myMathに代入します。

```
myMath = Math.atan2(y,x);
```
変数x,y からy/x のアーク・タンジェントを求めて変数myMathに代入します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS5	Android
	○	○	○	○	○	○	○	○	○

使用できる数値の範囲を調べたい・・・・・・・・・・P.266

MATH.06

対数を求めたい

Math.E	自然対数の底eを参照
Math.LN10	10の自然対数を参照(log10)
Math.LN2	2の自然対数を参照(log2)
Math.LOG10E	eの常用対数を参照(loge)
Math.LOG2E	eの2を底とする対数を参照(log2e)
Math.log(★)	eを底とする対数を算出(log★)
Math.exp(★)	eのべき乗を算出
● = Math.pow(◆, ▲)	べき乗を算出(◆▲)

★……値もしくは数式
●……結果として得られる値
◆……数値または数式
▲……数値または数式

形式 プロパティ(Math.E、Math.LN10、Math.LN2、Math.LOG10E、Math.LOG2E)
メソッド(Math.log、Math.exp、Math.pow)

対数に関するプロパティやメソッドです。

Eプロパティ

自然対数の底の値を返します。値は約2.718です。

log、expメソッド

logメソッドは引数として与えた数の自然対数(底はe)、expメソッドはeのべき乗を返します。★が文字列だった場合、NaN(p.251)を返します。

powメソッド

◆を▲乗した値を返します。◆が文字列だった場合、NaNを返します。

文例

```
document.write(Math.E);
```
自然対数eの底を表示します。

```
myMath = Math.LN10;
```
10の自然対数を変数myMathに代入します。

```
myMath = Math.LN2;
```
2の自然対数を変数myMathに代入します。

```
myMath = Math.LOG10E;
```
eの常用対数を変数myMathに代入します。

```
myMath = Math.LOG2E;
```
eの2を底とする対数を変数myMathに代入します。

```
myMath = Math.log(10);
```
10の自然対数を求め、変数myMathに代入します。

```
myMath = Math.exp(5);
```
eの5乗を求め、変数myMathに代入します。

```
document.write(Math.pow(3,i));
```
3のi乗を求め、表示します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

使用できる数値の範囲を調べたい・・・・・・・・・・P.266
【SAMPLE】対数、平方根、べき乗を算出する・・P.270

MATH.07

数値の大小を比較したい

● = Math.max(★,★, ...) 最大の数値を得る

● = Math.min(★,★, ...) 最小の数値を得る

●……結果として得られる文字列

★……数値または数式

式 メソッド

引数に指定した複数の数値の大小を比較します。maxメソッドは一番大きい数値、minメソッドは一番小さい数値を返します。★が文字列の場合、NaN(p.251)を返します。引数が与えられない場合はInfinityを返します。

文例

```
myMath = Math.max(8, 20);
```

8と20を比較し、大きい方の値を変数myMathに代入します(値は20)。

```
myMath = Math.min(a, b);
```

aとbを比較し、小さい方の値を変数myMathに代入します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

参照

使用できる数値の範囲を調べたい……P.266
【SAMPLE】ランダムに表示した2つの数字を比較する……P.267

MATH.08

平方根を求めたい

● = Math.sqrt(★)　平方根を算出(√★)
● = Math.SQRT2　2の平方根を参照(√2)
● = Math.SQRT1_2　2の平方根の半分の値を参照(√2 /2)

●……結果として得られる値
★……数値または数式

形式 メソッド(Math.sqrt)、プロパティ(Math.SQRT2、Math.SQRT1_2)

平方根を求めるメソッドやプロパティです。

sqrtメソッド

引数に指定した数値の平方根を返すメソッドです。★が文字列の場合、NaN(p.275)を返します。

SQRT2プロパティ

2の平方根の値(1.4142135623730951)を表すプロパティです。

SQRT1_2プロパティ

2の平方根の半分の値(0.7071067811865476)を表すプロパティです。

文例

```
myMath = Math.sqrt(num);
```

変数num の平方根を求め、変数myMathに代入します。

```
myMath = Math.SQRT2;
```

2の平方根を変数myMathに代入します。

```
myMath = Math.SQRT1_2;
```

2の平方根の半分の値を変数myMathに代入します。

▶ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照

使用できる数値の範囲を調べたい・・・・・・・・・・P.266
[SAMPLE] 対数、平方根、べき乗を算出する・・P.270

MATH.09

使用できる数値の範囲を調べたい

Number.MAX_VALUE	使用可能な最大値を参照
Number.MIN_VALUE	使用可能な最小値を参照
Number.NaN	[illegible]値以外を参照
Number.NEGATIVE_INFINITY	[illegible]の無限大の値を参照
Number.POSITIVE_INFINITY	無限大の[illegible]を参照

形式 プロパティ

JavaScriptで使用可能な数値の最大値や最小値などを保持しているプロパティです。これらのプロパティを利用すれば、使用できる数値の範囲を調べたり、数値がJavaScriptで扱える範囲なのかを調べることができます。

NaNは数値以外であることを表す特別な値で、Mathオブジェクトの各メソッドの引数に数値や数式以外を指定した場合に返されます。

文例

```
alert(Number.MAX_VALUE);
```
使用可能な最大値をダイアログに表示します。
```
alert(Number.MIN_VALUE);
```
使用可能な最小値をダイアログに表示します。
```
document.write(isNaN(myNum .NaN));
```
数値オブジェクトmyNun が数値かどうかを調べ、表示します(結果はtrue)。
```
alert(Number.NEGATIVE_INFINITY);
```
使用可能な負の最大値をダイアログに表示します。
```
alert(POSITIVE_INFINITY);
```
使用可能な無限大の値をダイアログに表示します。

Column Numberオブジェクト

Numberは数値を扱うオブジェクトです。JavaScriptで使用可能な最大値、最小値、無限大などを保持しています。

数値かどうかを調べたい ················ P.251

MATH.SAMPLE-01

ランダムに表示した2つの数字を比較する

現在表示されている1～13の数字と、次にランダムに表示される1～13の数字との大小を当てるゲームのサンプルです。まず、randomメソッドを使用して乱数を取得します。1～13までの数字をランダムに取得するには、乱数を13を乗じ1を加えます。

[HIを選択]、[LOWを選択]のどちらかのボタンがクリックされると新たな乱数を作成します。続いて、2つのランダムな数値をmaxメソッドまたはminメソッドで比較し、結果を表示します。

JavaScript

```
var num1;
var num2;
var numElem;
var bElem;
var hiElem;
var lowElem;

//ページ読み込み時の処理の
function loadPage(){
    numElem = document.getElementById("num");
    bElem = document.getElementById("b");
    hiElem = document.getElementById("btnHI");
    lowElem = document.getElementById("btnLOW");
    bElem.innerHTML = "HI または LOW を選択して下さい";
    num1 = Math.floor(Math.random() * 13) + 1; //1～13までのランダムな数字を作成
    numElem.innerHTML = num1;
    hiElem.disabled = false;
    lowElem.disabled = false;
}

//"HI"または"LOW"のボタンをクリックしたときの処理の
function clickBtn(value){
    var rBtn = "<input type='button' id='retry' value='リトライ' onclick='loadPage()' />"
    var win = "あなたの勝ちです" + rBtn;
    var lost = "あなたの負けです" + rBtn;
    num2 = Math.floor(Math.random() * 13) + 1;
    if(num1 == num2){                //同数の場合
        bElem.innerHTML = lost;
        numElem.innerHTML += " = " + num2;
```

```
    }else if(num2 == Math.min(num1, num2)){    //LOWの場合
        if(value == "hi"){
            bElem.innerHTML = win;
            numElem.innerHTML += " > " + num2;
        }else{
            bElem.innerHTML = lost;
            numElem.innerHTML += " < " + num2;
        }
    }else if(num2 == Math.max(num1, num2)){    //HIの場合
        if(value == "low"){
            bElem.innerHTML = win;
            numElem.innerHTML += " < " + num2;
        }else{
            bElem.innerHTML = lost;
            numElem.innerHTML += " > " + num2;
        }
    }
    hiElem.disabled = true;
    lowElem.disabled = true;
}
```

HTML

※レイアウトは外部CSSで指定しています

```
<body onload="loadPage()">
    <div>
        <b id="b"></b>
    </div>
    <div id="num">
    </div>
    <div>
        <input type="button" id="btnHI" value="HIを選択" onclick="clickBtn('hi')" />
        <input type="button" id="btnLOW" value="LOWを選択"
onclick="clickBtn('low')" />
    </div>
</body>
```

Internet Explorer

❶1～13の数字がランダムに表示されます。次にランダムに表示される1～13の数字を予想しましょう。次に表示される数字よりも現在表示されている13のほうが大きいと予想した場合は[HIを選択]ボタンをクリックします

❷ランダムに選ばれた値は11でした。13のほうが大きな値なので、「あなたの勝ちです」と表示されます

参照
random メソッド …… P.256
floor メソッド …… P.257
max メソッド …… P.264
min メソッド …… P.264

MATH.SAMPLE-02

対数、平方根、べき乗を算出する

対数、平方根、べき乗を算出するサンプルです。2を10乗した値、4の平方根、2の平方根の半分の値、2の値、自然対数の値などを表示します。

JavaScript

```
document.write("<b>2の10乗= </b>", Math.pow(2, 10), "<br />");
document.write("<b>√4= </b>", Math.sqrt(4), "<br />");
document.write("<b>√2の1/2= </b>", Math.SQRT1_2, "<br />");
document.write("<b>√2= </b>", Math.SQRT2,"<br />");
document.write("<b>E= </b>", Math.E, "<br />");
document.write("<b>LN10=</b>", Math.LN10, "<br />");
document.write("<b>LN2= </b>", Math.LN2, "<br />");
document.write("<b>LOG10E= </b>", Math.LOG10E, "<br />");
document.write("<b>LOG2E= </b>", Math.LOG2E, "<br />");
document.write("<b>log(100)= </b>", Math.log(100), "<br />");
document.write("<b>exp(100)= </b>", Math.exp(100));
```

参照

E プロパティ ・・・・・・ P.262
LN10 プロパティ ・・・・・・ P.262
LN2 プロパティ ・・・・・・ P.262
LOG10E プロパティ ・・・・・・ P.262
LOG2E プロパティ ・・・・・・ P.262
log メソッド ・・・・・・ P.262
exp メソッド ・・・・・・ P.262
pow メソッド ・・・・・・ P.262
sqrt メソッド ・・・・・・ P.265
SQRT2 プロパティ ・・・・・・ P.265
SQRT1_2 プロパティ ・・・・・・ P.265

MATH.SAMPLE-03

使用できる数値の範囲を調べる

数値オブジェクトが持つプロパティの値を表示するサンプルです。演算などを行う上で、JavaScriptで使用できる最大値や最小値を参照します。

JavaScript

```
document.write("<b>MAX_VALUE =</b>", Number.MAX_VALUE, "<br />");
document.write("<b>MIN_VALUE =</b>", Number.MIN_VALUE, "<br />");
document.write("<b>NaN =</b>", Number.NaN, "<br />");
document.write("<b>NEGATIVE_INFINITY =</b>", Number.NEGATIVE_INFINITY,
"<br />");
document.write("<b>POSITIVE_INFINITY =</b>", Number.POSITIVE_INFINITY);
```

MAX_VALUE プロパティ・・・・・P.266
MIN_VALUE プロパティ・・・・・P.266
NaN プロパティ・・・・・P.266
NEGATIVE_INFINITY プロパティ・・・・・P.266
POSITIVE_INFINITY プロパティ・・・・・P.266

OBJECT.01

独自のオブジェクトを使いたい

★ = new Object(◆)

★……新しいオブジェクトを格納する変数
◆……値【[illegible]可】

形式 オブジェクト

新しいオブジェクトを生成するには、newステートメント（p.037）を使用します。JavaScriptの仕様としてあらかじめ用意されているオブジェクト以外に独自にオブジェクトを作成したい場合には、このようにnewステートメントを使用します。

オブジェクト作成時の引数として指定する値には数値、論理値、文字列、関数といった基本型のオブジェクトを指定できます。また、引[illegible]を指定した場合、オブジェクトを参照すると、プロパティのない新しく作成したオブジェクトが返ります。引数を指定し、オブジェクトを参照した際は引数で指定したオブジェクトを返します。

文例

```
newObj = new Object();
```

新規にObject オブジェクトnewObj を生成します。

▶ ブラ[illegible]対応表	IE10	IE9	IE8	Fx	Chrome	Safari	[illegible]	[illegible]	Android
	○	○	○	○	○	○	○	○	○

参照
（基礎編）オブジェクトを扱う ………… P.037
【SAMPLE】オブジェクトを扱う ………… P.278

OBJECT.02

オブジェクトのコンストラクタや値を参照したい

★.constructor	オブジェクトのコンストラクタを参照
◆ = ★.valueOf()	オブジェクトの値を返す

★……オブジェクト名　　　▲……■

形式 プロパティ(constructor)、メソッド(valueOf)

オブジェクトのコンストラクタ(生成関数：p.284)の値を参照します。

constructorプロパティ

そのオブジェクトを作成するときに実行されるコンストラクタを返すプロパティです。コンストラクタの内容を知りたいときはtoSourceメソッド(p.274)を使います。

valueOfメソッド

オブジェクトの値を返すメソッドです。返す値は下表の通りです。

オブジェクト	valueOfメソッドの動作
Array	Array.toStringメソッドやArray.joinメソッドと同じような値を返す
Boolean	Objectに格納されているブール値を返す
Date	協定世界時の1970年1月0日0時0分0秒からの経過時間を表すミリ秒単位の数値を返す
Function	関数自体を返す
Number	オブジェクトに格納されている数値を返す
Object	オブジェクト自体を返す
String	オブジェクトに格納されている文字列を返す

直接オブジェクトを参照すれば、オブジェクトの値が返るため、普段参照する際には使わなくても構いません。下の文例の2番目はdocument.write(newObj);としても同じです。

文例

```
alert(newObj.constructor);
```

newObj オブジェクトのコンストラクタをダイアログに表示します。

```
document.write(newObj.valueOf());
```

newObj オブジェクトの値を書き出します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

参照
プログラムの内容を知りたい・・・・・・・・・・・・・・P.274
関数呼び出しの情報を調べたい・・・・・・・・・・・P.281
【SAMPLE】オブジェクトを扱う・・・・・・・・・・P.278

OBJECT.03

プログラムの内容を知りたい

★.toSource()

★……[illegible]やオブジェクト名

形式 メソッド

指定した関数やオブジェクトの内容を表示します。得られる値についてはサンプル(p.275)を参照してください。

文例

```
document.write(newObj.toSource());
```

newObj オブジェクトのスクリプトの内容を書き出します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	[illegible]	[illegible]	Android
	×	×	×	○	○	×	×	×	×

参照
オブジェクトのコンストラクタの■を参照したい ……………… P.273
【SAMPLE】プログラムの内容を表示する …… P.275

OBJECT.SAMPLE-01

プログラムの内容を表示する

文字列オブジェクト、日付オブジェクト、ブール型オブジェクト、配列オブジェクトおよび関数の内容を取得し、表示するサンプルです。値に加えてtoSourceメソッドを使用して各オブジェクト、関数の内容を取得し、表示させています。Internet Explorer、OperaはtoSourceメソッドに対応していないため、処理を分岐させ「参照できません」と表示するようにしています。

JavaScript

```
//変数の宣言
var newStr = new Object("JavaScript辞典");
var newDate = new Date(document.lastModified);
var newBool = new Boolean(true);
var newArray = new Array("ホームページ辞典","HTMLタグ辞典","スタイルシート辞典");
var boolIE = navigator.appName.indexOf("Microsoft")<0;
var boolOP = navigator.appName.indexOf("Opera")<0;
var html = "";

//各オブジェクトの情報を参照して表示する関数
function loadPage(){
  //文字列オブジェクトの参照
  html += "<b>文字列オブジェクト：</b>" + newStr + "<br />";
  html += "<b>オブジェクトの値：</b>" + newStr.valueOf() + "<br />";
  if(boolIE && boolOP){
    html += "<b>オブジェクトの内容：</b>" + newStr.toSource() + "<hr />";
  }else{
    html += "<b>オブジェクトの内容：</b>参照できません<hr />";
  }
  //日付列オブジェクトの参照
  html += "<b>日付オブジェクト：</b>" + newDate + "<br />";
  html += "<b>オブジェクトの値：</b>" + newDate.valueOf() + "<br />";
  if(boolIE && boolOP){
    html += "<b>オブジェクトの内容：</b>" + newDate.toSource() + "<hr />";
  }else{
    html += "<b>オブジェクトの内容：</b>参照できません<hr />";
  }
  //ブール型オブジェクトの参照
  html += "<b>ブール型オブジェクト：</b>" + newBool + "<br />";
  html += "<b>オブジェクトの値：</b>" + newBool.valueOf() + "<br />";
```

```
    if(boolIE && boolOP){
        html += "<b>オブジェクトの内容：</b>" + newBool.toSource() + "<hr />";
    }else{
        html += "<b>オブジェクトの内容：</b>参照できません<hr />";
    }
    //配列オブジェクトの参照
    html += "<b>配列オブジェクト：</b>" + newArray + "<br />";
    html += "<b>オブジェクトの値：</b>" + newArray.valueOf() + "<br />";
    if(boolIE && boolOP){
        html += "<b>オブジェクトの内容：</b>" + newArray.toSource() + "<hr />";

    }else{
        html += "<b>オブジェクトの内容：</b>参照できません<hr />";

    }
    //関数オブジェクトの参照
    if(boolIE && boolOP){
        html += "<b>関数の内容：</b>" + fSample.toSource() + "<hr />";
    }else{
        html += "<b>関数の内容：</b>参照できません<hr />";
    }
    document.getElementById("body").innerHTML = html;
}

//確認するためのサンプル関数
function fSample(){
    newArray = new Array("ホームページ辞典","HTMLタグ辞典","スタイルシート辞典");
    html += "配列オブジェクト：" + newArray;
    html += "オブジェクトの値：" + newArray.valueOf();
    html += "オブジェクトの内容：" + newArray.toSource();
}
```

HTML

```
<body id="body" onload="loadPage()">
</body>
```

Internet Explorer

Internet ExplorerはtoSourceメソッドに対応していないので、オブジェクトの内容には「参照できません」と表示されます

Firefox

オブジェクトや関数の内容が書き出されます

toSource メソッド ・・・・・・・・・・・・・・・・・・ P.274

OBJECT.SAMPLE-02

オブジェクトを扱う

各種オブジェクトの作成元、値を表示するサンプルです。文字列オブジェクト、日付オブジェクト、配列オブジェクトを作成して、それぞれ作成元のオブジェクトとオブジェクトの値を参照しています。

JavaScript

```
//変数の宣言
var newObj = new Object("JavaScript辞典");
var newDate = new Date(document.lastModified);
var newBool = new Boolean(true);
var newArray = new Array("ホームページ辞典","HTMLタグ辞典","スタイルシート辞典");
//文字列オブジェクトの参照
document.write("<b>文字列オブジェクト:</b>", newObj, "<br />");
document.write("<b>作成元のコンストラクタ:</b>", newObj.constructor, "<br
/>");
document.write("<b>オブジェクトの値:</b>", newObj.valueOf(), "<hr />");
//日付列オブジェクトの参照
document.write("<b>日付オブジェクト:</b>", newDate, "<br />");
document.write("<b>作成元のコンストラクタ:</b>", newDate.constructor, "<br
/>");
document.write("<b>オブジェクトの値:</b>", newDate.valueOf(), "<hr />");
//ブール型オブジェクトの参照
document.write("<b>ブール型オブジェクト:</b>", newBool, "<br />");
document.write("<b>作成元のコンストラクタ:</b>", newBool.constructor, "<br
/>");
document.write("<b>オブジェクトの値:</b>", newBool.valueOf(), "<hr />");
//配列オブジェクトの参照
document.write("<b>配列オブジェクト:</b>", newArray, "<br />");
document.write("<b>作成元のコンストラクタ:</b>", newArray.constructor, "<br
/>");
document.write("<b>オブジェクトの値:</b>", newArray.valueOf());
```

Internet Explorer

iphone

参照
Object オブジェクト …………………… P.272
constructor プロパティ ……………… P.273
valueOf メソッド …………………… P.273

FUNCTION.01

関数を作成したい

★ = new **Function**(◆,◆,...,◆,▲)

★……Functionオブジェクトを格納する変数
◆……引数【省略可】
▲……処理の内容

形式 オブジェクト

Functionは関数を扱うオブジェクトです。新しくFunctionオブジェクトを作成するにはnewステートメント(p.037)を使用しますが、通常は明示的に作成することはありません。

動作は通常のfunction{ }による関数定義と同じです。★で指定した変数の名前が関数名になります。

文例

```
msg = new Function("alert(arguments[0] + arguments[1])");
```

最初の引数と2番目の引数を結合して、ダイアログに表示する関数msgを作成します。
なお、明示的に関数オブジェクトを作成せず、

```
function msg() {
alert(arguments[0] + arguments[1]);
}
```

としても同様の効果を得られます。

```
myCalc = new Function(i, j,"return i+j");
```

引数i、引数jを加算した結果を返す関数myCalcを作成します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照 (基礎編) 関数 ·························· P.039

FUNCTION.02

関数呼び出しの情報を調べたい

arguments.length	呼び出し時に実際に与えられた引数の数を参照
★.length	関数が期待する、関数定義時の引数の数を参照
★.caller	呼び出し元の関数オブジェクトを参照

★……Functionオブジェクト（関数名）

形式 プロパティ

argumentsは関数に渡された引数を格納した配列です。その関数内のローカル変数としてもFunctionオブジェクトのプロパティとしても参照できますが、後者は非推奨です。

lengthプロパティ

定義上の引数の数を参照します。実際に与えられた引数の数はarguments.lengthです。

callerプロパティ

この関数★を呼び出している関数（Functionオブジェクト）を参照します。トップレベルで関数★が呼び出されているときはnullを返します。

文例

n = **arguments.length**
関数newFuncの引数の数を変数nに格納します（関数newFunc内の記述）。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

オブジェクトのコンストラクタの値を参照したい・・P.273
関数内からほかの関数を呼び出したい・・・・・・P.282
【SAMPLE】関数呼び出し情報を調べる・・・・・・P.285

FUNCTION.03

関数内からほかの関数を呼び出したい

★.apply(▲, arguments)
★.call(▲,◆,◆...)

★……呼び出すFunctionオブジェクト(関数名)
▲……呼び出された関数内でthisに格納されるオブジェクト
◆……引数

式 メソッド

applyメソッドおよびcallメソッドは、関数内からほかの関数を呼び出します。applyとcallメソッドの違いは引数だけで、動作は同じです。

applyメソッド

第1引数に、呼び出す関数内でthisとして表したいオブジェクトを指定します。第2引数にはargumentsを指定し、まとめて引数を渡します。

callメソッド

第1引数に、呼び出す関数内でthisとして表したい呼び出し側のオブジェクトを指定します。第2引数以降には、引数として渡す変数を個々に指定します。

文例

```
calc.apply(this, arguments);
```

関数calcを呼び出します。このとき、これを実行している関数で定義された引数をすべて関数calcに渡します。

```
calc.call(this, a, b);
```

関数calcを呼び出します。このとき、これを実行している関数で定義された引数a,bを関数calcに渡します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

関数呼び出しの情報を調べたい ………… P.281
【SAMPLE】関数からほかの関数を呼び出す … P.283

FUNCTION.SAMPLE-01

関数内からほかの関数を呼び出す

［半径］の値が変更されるとcalc関数が呼び出されます。関数内では、Circleオブジェクトを作成し、そのcircumference属性を円周、area属性を面積として画面に表示させています。

Circle関数内では、生成したCircleオブジェクトの属性をsetPI関数を使って初期化しています。setPI関数は、this変数の指定された属性に、引数value×円周率の結果を格納する関数です。通常の関数呼び出しではこのthisはwindowオブジェクトですが、applyメソッドを利用すると、第1引数でこのthisの中身を指定することができます。

JavaScript

```
function calc() { // 半径を[illegible]関数
    var radius = document.getElementById("radius").value;
    var circle = new Circle(radius); // [illegible]を作成
    document.getElementById("circum").value = circle.circumference;
    document.getElementById("area").value = circle.area;
}
function Circle(radius) { [illegible]
    setPI.apply(this, ["circumference", radius * 2]); [illegible]
    setPI.apply(this, ["area", radius * radius]); [illegible]
}
function setPI(prop, value) { // [illegible]
    // [illegible]
    // [illegible]
    this[prop] = value * Math.PI;
}
```

HTML

```
<body>
<b>円の半径を入力すると、円周と面積を算出します</b>
<form action="">
    <p>
        半径：<input type="text" id="radius" size="30" onchange="calc()" />
    </p>
    <p>
        円周：<input type="text" id="circum" size="30" onchange="calc()"
/><br/>
        面積：<input type="text" id="area" size="30" onchange="calc()" />
    </p>
```

```
</form>
</body>
```

Internet Explorer

半径を入力し、テキスト入力フィールドからフォーカスを離すと、円周と面積が計算されます

Column　ユーザー定義オブジェクト

任意の関数をnewを使って呼び出すと、その関数が生成したオブジェクトが返されます(returnの値は無視されます)。この際、生成した関数をコンストラクタと呼び、コンストラクタを独自に定義することで、独自のオブジェクトを生成することもできます。

```
Studentオブジェクト = new Student(任意の数の引数);
```

newで呼び出されると、関数内のthis変数には、生成されたオブジェクトが格納されます(p.037)。このthisを利用することで、オブジェクトの初期化を行います。

```
function Student (a) {
        this.name = a;
}
```

上記ではStudentオブジェクトを生成してnameプロパティを定義しています。メソッドについては、関数の外で、以下のような書式で定義します。

```
Student.prototype.hello = function(任意の引数){メソッドの処理};
```

この例ではStudentオブジェクトにhelloメソッドを定義しています。メソッドとして呼び出された場合、その関数内のthisは、そのメソッドが所属しているオブジェクトになります。

apply メソッド ・・・・・・・・・・・・・・・・・・・・・ P.282

FUNCTION.SAMPLE-02

関数呼び出しの情報を調べる

関数の情報を表示するサンプルです。ページが読み込まれるとloadPage関数を呼び出し、配列オブジェクトの作成とargCheck関数を呼び出しを行います。argCheck関数では関数の情報を取得して表示しています。

JavaScript

```
function loadPage() { //配列の作成とargCheck関数を呼び出す関数
   argCheck.call(this, "1", "2", "3", "4", "5", "6");
}
function argCheck(a, b, c, d) { //関数の呼び出し情報を参照して表示する関数
   document.write("<b>argCheck関数の情報</b><hr />");
   document.write("<b>実際の引数の数(arguments):</b>", arguments.length,
"<br />");
   document.write("<b>本来の引数の数(length):</b>", argCheck.length, "<br
/>");
   document.write("<b>呼び出し元:</b>", argCheck.caller, "()<br />");
}
```

HTML

```
<body onload="loadPage()">
</body>
```

length プロパティ ･････････････････････ P.281
caller プロパティ ･････････････････････ P.281
constructor プロパティ ･････････････････ P.281

REGULAR EXPRESSION.01

正規表現を使いたい

★ = new RegExp(◆,▲)

正規表現オブジェクトを作成

★……RegExpオブジェクトを格納する変数
◆……パターン(正規表現を文字列で表したもの)
▲……フラグ【省略可】

関連 オブジェクト

正規表現(Regular Expression)は、一定のルールに従って文字列の規則的なパターンを表現したもので、検索や置換の際に使用します。

正規表現を使用するには、RegExpオブジェクトを作成します。引数の◆は正規表現の文字列、▲はフラグです。フラグにはgとiがあり、gを指定した場合には一致がすべて検索されます。iを指定した場合は大文字と小文字の区別を無視します。

検索は、RegExpオブジェクトのexecメソッド(p.291)で実行します。結果はオブジェクトの属性$1～$9に格納されます。また、Stringオブジェクトのmatch、search、replaceメソッドの引数にRegExpオブジェクトを指定して、検索や置換を行うこともできます(p.293)。

なお、★=new RegExp(◆,▲)の書式は★=/◆/▲としても同じ結果になります。

Column　パターンの設定方法

RegExpオブジェクトの作成時やcompileメソッドで設定するパターンは、さまざまな種類があります。また、以下の2行は同じ意味になります。

```
re = new RegExp("abc", "i"); // newを使ったRegExpオブジェクトの初期化
re = /abc/i; // 「/」を使ったRegExpオブジェクトの初期化
```

パターンの使用例をいくつか紹介します。

改行文字以外の任意の文字と一致

scとそれに続く改行文字以外の任意の2文字を検索(大文字小文字は区別なし)。

```
result = "JavaScript辞典".match(/sc../i); // 一致結果は「Scri」
```

角カッコ内のいずれかの文字と一致

x、s、dのうちのどれか1文字とそれにcが続く部分を検索(大文字小文字は区別なし)。

```
result = "JavaScript辞典".match(/[xsd]c/i); // 一致結果は「Sc」
```

どちらかの文字と一致

JavaまたはVBとそれに続くscriptの部分を検索(大文字小文字の区別はなしで一致するすべての部分を検索)。変数resultにはJavaScriptが格納されます。

```
result = "JavaScript辞典".match(/(Java|VB)script/gi);
```

[illegible]	意味	[illegible]	[illegible]
¥	¥に続く文字は特別な文字またはリテラル¥n		
¥(			
改行文字			
(			
^	入力の開始と一致^1	1で始まる行	
$	入力の終端と一致0$	0で終わる行	
*	直前の文字と0回以上一致1a*	1、1a、1aa、1aaaなど	
+	直前の文字と1回以上一致1a+	1a、1aa、1aaaなど	
?	直前の文字と0回または1回一致books	?	book、books
.	改行文字以外の任意の1文字と一致	.ap	map、cap、japなど
(★)	()をグループとみなす	(ca)+tp	cap、cacapなど
★|◆	★と◆のどちらかと一致	(c|m)ap	cap、map
{★}	直前の文字と正確に★回一致	zo{2}	zoo
{★,}	直前の文字と最低でも★回一致	zo{2,}	zoo、zooo、zooooなど
{★,◆}	直前の文字と★~◆回一致	zo{2,3}	zoo、zooo
[★◆▲...]	角かっこで囲まれた文字の中のいずれかと一致	[lm]ake	lake、make
[^★◆▲...]	角かっこで囲まれた文字にはない任意の文字と一致	[^ lm]ake	cakeなど（lake、make以外）
[★-◆]	指定した範囲に含まれる任意の文字に一致	[1-3]人	1人、2人、3人
[^★-◆]	指定した範囲に含まれていない任意の文字に一致	[^1-3]人	4人、5人など（1人、2人、3人以外）
¥b	単語とスペースの間の位置に一致	¥ba	aで始まる単語
¥B	単語とスペースの間の位置ではない部分と一致		
¥d	数字と一致。[0-9]と指定した場合と同じ	¥d体	1体、2体など
¥D	数字以外の文字と一致。[^0-9]と指定した場合と同じ	¥D体	身体、固体など
¥f	フォームフィード文字と一致		
¥n	改行文字と一致	¥nThe	Theで始まる段落
¥r	キャリッジリターン文字と一致		
¥s	スペース、タブ、フォームフィードなどの任意の空白文字と一致		
¥S	空白文字以外の部分と一致		
¥t	タブ文字と一致		
¥v	垂直タブ文字と一致		
¥w	単語に使用される任意の文字（アルファベットの大文字と小文字、アンダースコアおよび数字）と一[illegible]		
¥W	単語に使用される文字以外の任意の文字と一致		
¥★	すでに見つかり、記憶されている部分と一致		
¥★	★に指定した8進数のエスケープ値と一致。8進数の値は、1桁、2桁、または3桁で指定します。256を超える286287数値を指定した場合、初めの2桁で値が評価される		
¥x★	★に指定した16進数のエスケープ値と一致。16進数のエスケープ値は、2桁で指定		

▶ ブラウザ対応表	IE10	IE9	IE8	[illegible]	[illegible]	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

（基礎編）オブジェクトを扱う・・・・・・・・・・・・・P.037
【SAMPLE】テキストを正規表現検索する・・・・P.295

REGULAR EXPRESSION.02

正規表現のオプションを調べたい

★.global	完全一致検索かどうか参照
★.ignoreCase	大文字と小文字の区別を参照
RegExp.multiline	改行コードを無視するかどうかを参照／設定

★……RegExpオブジェクト

形式 プロパティ

RegExpオブジェクトの現在の設定を参照します。

global、ignoreCaseプロパティ

RegExpオブジェクト作成時に指定するフラグ（gまたはi）の設定状態を返します。globalプロパティの値がtrueの場合（フラグがgのとき）、最初の一致だけでなくすべての一致を検索し、結果を配列に格納します。ignoreCaseプロパティの値がtrueの場合（フラグがiのとき）は、検索時に大文字と小文字を区別しなくなります。それぞれ設定されていない場合はfalseを返します。これらのプロパティでは、設定の変更はできません。設定後にフラグを変更したい場合は、compileメソッド（p.291）を使用して新しい設定を適用してください。

multilineプロパティ

検索時に改行コードを無視するかどうかを参照／設定するプロパティです。値はtrue（改行を無視しない）またはfalse（改行を無視する）になります。multilineプロパティは$*で置き換えることも可能です。

文例

```
alert(matchStr.global);
```
RegExp オブジェクトmatchStr が完全一致検索かどうかを表示します。

```
alert(matchStr.ignoreCase);
```
RegExp オブジェクトmatchStr が大文字と小文字を区別しているかどうかを表示します。

```
RegExp.multiline = false;
```
改行コードを無視し複数行に対して検索するように指定します。

ブラウザ対応表	IE10	IE9	IE8	IE7	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照
正規表現で文字列を検索／置換したい ……P.293
【SAMPLE】テキストを正規表現検索する ……P.295

最後に一致する文字列を参照したい

RegExp.lastMatch　最後に一致した文字列を参照
RegExp.lastParen　最後に一致したグループの文字列を参照

形式 プロパティ

一致した文字列を参照するプロパティです。

lastMatchプロパティ

最後に一致した文字列を参照します。lastMatchは$&で置き換えることができます。つまり、RegExp.lastMatchとRegExp.$&は同じ結果を返します。

lastParenプロパティ

最後にマッチした文字のうち、検索パターンの最後のグループ(()でくくられた部分)に相当する文字列を参照します。lastParenは$+で置き換えることができます。

文例

```
alert(RegExp.lastMatch);
```
最後に一致した文字列をダイアログに表示します。

```
document.write(RegExp.lastParen);
```
最後に一致したグループの文字列を書き出します。

ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

一致する文字列の左右の文字列を参照したい‥P.290
正規表現で文字列を検索／置換したい‥‥‥P.293

REGULAR EXPRESSION.04

一致する文字列の左右の文字列を参照したい

RegExp.leftContext 最後に一致した文字列より前の文字列を参照
RegExp.rightContext 最後に一致した文字列より後ろの文字列を参照

形式 プロパティ

一致する文字列の左右の文字列を参照するプロパティです。

leftContextプロパティ

最後に一致した文字列より前の文字列を参照します。leftContextは$`で置き換えることができます。

rightContextプロパティ

最後に一致した文字列より後ろの文字列を参照します。rightContextは$'で置き換えることができます。

文例

```
matchStr = RegExp.leftContext;
```

最後に一致した文字列より前の文字列を変数matchStr に代入します。

```
document.write(RegExp.rightContext);
```

最後に一致した文字列より後ろの文字列を書き出します。

ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

最後に一致する文字列を参照したい ········ P.289
正規表現で文字列を検索／置換したい ······ P.293

正規表現の文字列を変更したい

☆ = ★.compile(◆, ▲)	パターン文字列を設定または変更
RegExp.input	検索する文字列を参照／設定
★.source	パターン文字列を参照
▼ = ★.test(●)	一致する文字列が含まれているかどうかを調べる
■ = ★.exec(●)	検索を実行
★.lastIndex	検索の開始位置を参照／設定

☆……RegExpオブジェクト
★……RegExpオブジェクト
◆……パターン
▲……フラグ【省略可】
▼……trueまたはfalse
●……検索の対象となる文字列
■……結果が格納される変数

形式 メソッド(compile、input、test、exec)
プロパティ(source、lastIndex)

compile、inputメソッド

compileメソッドでは、検索に使用するパターン文字列を設定または変更します。検索対象の文字列はinputメソッドで設定できます。これらのメソッドを使用すると、ユーザーがフォームに入力したパターンや検索対象文字列などを使って検索できます。

sourceプロパティ

newステートメントやcompileメソッドによって設定されたパターン文字列を参照できます。

testメソッド

◆と一致する文字列が含まれているかどうかのみを調べます。結果として、一致する文字列があればtrue、なければfalseを返します。

execメソッド

検索を実行するするメソッドです(matchメソッドでも可能。p.293)。パターン文字列に一致する文字列が見つかった場合は結果の配列を返し、見つからなかった場合はnullを返します。検索の対象となる文字列を省略した場合、RegExp.inputで設定された文字列が検索されます。

lastIndexプロパティ

検査の開始位置を指定します。たとえばlastIndexプロパティを5に設定した場合、それより前の部分(1～5文字目)に一致する文字列があっても検出されません。

文例

```
matchStr = re.compile("julia");
```

RegExpオブジェクトreのパターン文字列をjuliaに変更したRegExpオブジェクトmatchStrを作ります。

```
RegExp.input="ism";
```

検索文字列をismに設定します。

```
alert(re.source);
```

RegExpオブジェクトreのパターン文字列をダイアログに表示します。

```
testResult =re.test(str);
```

RegExpオブジェクトreと文字列str のパターンマッチを行い、結果を変数testResultに代入します。

```
execResult = re.exec(pat);
```

RegExpオブジェクトreと文字列strのパターンマッチを行い、結果を変数execResultに代入します。

```
re.lastIndex = 4;
```

RegExpオブジェクトreを5番目の文字から検索するように設定します。

ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照
正規表現で文字列を検索／置換したい ･･････ P.293
正規表現のオプションを調べたい･･････････ P.288

正規表現で文字列を検索／置換したい

● = ★.match(◆)　検索し、一致した文字列を返す
■ = ★.search(◆)　検索し、一致した位置を返す
▼ = ★.replace(◆, ▲)　置換する

●……一致した文字列
★……Stringオブジェクト（文字列）
◆……正規表現文字列
■……一致した位置
▼……置換された文字列
▲……置換文字列

形式 メソッド

正規表現を使用して文字列の検索や置換を行います。★には文字列または文字列が格納された変数を指定します。正規表現文字列◆には、RegExpオブジェクトのメソッドやプロパティで設定したパターン文字列を指定します。

matchメソッド

文字列の検索を行います。一致した場合はその文字列を返し、一致する文字列がなかった場合はnullを返します。

searchメソッド

文字列の検索を行います。一致した場合、一致した文字列の先頭からの位置を返します。先頭文字の位置は0です。一致する文字列がなかった場合は-1を返します。

replaceメソッド

文字列の置換を行います。一致した場合はその文字を指定した文字に置き換え、一致する文字列がない場合はnullを返します。

文例

```
str.match(/xp/i);
```

文字列strからxpの文字を、大文字小文字を無視して検索します。

```
theResult = str.search(/xp/i);
```

文字列strからxpの文字を大文字小文字を無視して検索し、結果の位置を変数theResultに代入します。

```
theResult = str.replace("Sun.", "Sat.");
```

文字列strのSun.をSat.に置き換え、結果を変数theResultに代入します。

▶ [illegible]対応表	IE10	IE9	IE8	[illegible]	[illegible]	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

最後に一致する文字列を参照したい……… P.289
一致する文字列の左右の文字列を参照したい‥ P.290
【SAMPLE】テキストを正規表現検索する‥‥ P.295
【SAMPLE】正規表現で文字列を検索／置換する‥ 298

REGULAR EXPRESSION.SAMPLE-01

テキストを
正規表現検索する

テキスト■に含まれるテキストを正規表現検索するサンプルです。[検索開始]ボタンがクリックされるとsearcher関数を呼び出します。[検索する文字列]■の内容を引数として正規表現オブジェクトを作成し、matchメソッドを使用して検索を行っています。正規表現オブジェクトのフラグはラジオボタンでの切り替えが可能になっています。検索が終了したら、ダイアログでフラグの状態を表示し、検索結果は画面下段の「検索結果」テキストエリアに表示します。

JavaScript

```
//テキストを正規表現検索する関数
function searcher(){
    var flag = "";
    var formElem = document.getElementById("form1");
    if(formElem.global[1].checked)
        flag = "g";  //[すべて]がオンならフラグをg
    if(formElem.ignoure[1].checked){
        flag += "i";//[なし]がオンならフラグにiを加える
    }
    var str = formElem.text1.value;
    var textValue = formElem.textA1.value;
    var matchStr = new RegExp(str, flag);//正規表現オブジェクト作成
    var searchResult = textValue.match(matchStr);
                                        //正規表現での検索結果を変数に代入

    formElem.textA2.value = searchResult;
    alert("globalプロパティは"+ matchStr.global + "¥nignoreCaseプロパティは" +
matchStr.ignoreCase);
}
```

HTML

```
<body>
<form action="" id="form1">
    <p>
    検索する文字列(正規表現使用可能):
    <input type="text" name="text1" size="30" />
    <input type="button" name="btn1" value="検索開始" onclick="searcher()"
/><br/>
    検索対象:
    <input type="radio" name="global" checked="checked" />1つ目
    <input type="radio" name="global"/>すべて<br/>
    大文字、小文字の区別:
    <input type="radio" name="ignoure" checked="checked" />あり
    <input type="radio" name="ignoure"/>なし<br />
    </p>
    <hr />
    <p>
    検索対象となる文字列:<br />
    <textarea name="textA1" cols="80" rows="10">
  金井湛君は哲学が職業である。
  哲学者という概念には、何か書物を書いているということが伴う。
  (…中略…)
  真面目な講義の中で、その頃青年の読んでいる小説なんぞを引いて説明するので、学生がびっく
りすることがある。
    </textarea>
    </p>
    <hr />
    <p>
    検索結果:<br/>
    <textarea name="textA2" cols="80" rows="3"></textarea>
    </p>
</form>
</body>
```

Internet Explorer

検索する文字列(パターン)に「[あ-ん]哲学」を指定すると、ひらがな+哲学の3文字が検索されます

検索する文字列(パターン)に「..哲学」を指定すると、哲学で終わる4文字が検索されます

参照
RegExp オブジェクト …… P.286
global プロパティ …… P.288
match メソッド …… P.293
ignoreCase プロパティ …… P.288

REGULAR EXPRESSION.SAMPLE-02

正規表現で文字列を検索／置換する

入力と指定した正規表現との一致を調べるサンプルです。郵便番号、電話番号、メールアドレスのそれぞれの値について、matchメソッドを使ってチェックします。「郵便番号」は「●●●-●●●●」、「電話番号」は10桁の数字、「メールアドレス」は「○○○@○○○.●●.●●」、「○○○@○○○.●●●」、または「○○○@○○○.●●」の形式との一致をチェックしています。
※●の部分は文字数は固定、○の部分は1文字以上の数です。

JavaScript

```
function clickBtn(id) { //「確認」ボタンをクリックしたときの処理の関数
  switch (id) { // 渡されたidで処理を分岐
    case "post": check(id, "郵便番号", [/^¥d{3}[-]¥d{4}$/, null]);
      break;
    case "tel": check(id, "電話番号", [/^¥d{10}$/, null]);
      break;
    case "email": check(id, "メールアドレス", [
        /^¥w+([-]?¥w+)*@¥w+([-]?¥w+)*(¥.¥w{2,3}){1}$/,
        /^¥w+([-]?¥w+)*@¥w+([-]?¥w+)*(¥.¥w{2}){1}(¥.¥w{2}){1}$/]);
      break;
  }
}
function check(id, name, re) { // 実際のチェックを行う関数
  var el = document.getElementById(id);
  if (el.value.match(re[0]) || el.value.match(re[1])) {
    alert(name + "の形式になっています。");
  } else {
    alert(name + "を正確に入力して下さい。");
    el.focus();
    el.select();
  }
}
```

HTML

```
<body>
  <form action="">
    <p>
      郵便番号:<input type="text" id="post"/>
      <input type="button" value="確認" onclick="clickBtn('post')" />
    </p>
    <p>
      電話番号(ハイフンなし):<input type="text" id="tel"/>
      <input type="button" value="確認" onclick="clickBtn('tel')" />
    </p>
    <p>
      メールアドレス:<input type="text" id="email"/>
      <input type="button" value="確認" onclick="clickBtn('email')" />
    </p>
  </form>
</body>
```

Internet Explorer

入力した値が指定された正規表現と一致した場合

一致しなかった場合

参照 match メソッド ・・・・・・・・・・・・・・・・・・・・ P.293

DOM 01

オブジェクトの情報を取得したい

● = document.getElementById(★) idで取得
● = document.getElementsByTagName(◆) 要素名で取得
● = document.getElementsByName(▲) name属性値で取得

●……取得された要素
★……id名
◆……要素名
▲……name属性の値

書式 メソッド

DOM(Document Object Model)はスクリプトやプログラム言語からHTML / XHTML文書やXML文書にアクセスして操作するための規格で。W3Cによって標準化されています。DOMを利用すると、HTML / XHTMLの要素をすべてJavaScriptのオブジェクトとして操作できます。これらのオブジェクトはドキュメントにおいてツリー構造を構成しており、「ノード」と呼ばれます(要素ノード)。また、DOMではHTML / XHTMLの要素だけでなく属性や要素内容のテキストなどもノードとして扱われます(属性ノード、テキストノード)。

JavaScriptから要素を操作するためには要素ノードをオブジェクトとして取り出す必要があります。そのために次のようなメソッドがあります。

getElementByIdメソッド

指定したid(id属性の値)を持つ要素ノードを取得します。指定したidを持つ要素が見つからない場合はnullを返します。

getElementsByTagNameメソッド

指定した要素名の要素をすべて取り出し、配列として返します。このメソッドは要素オブジェクトのメソッドとして利用でき、その場合には要素の子孫要素のうちで、指定した要素名のノードリストを返します。

getElementsByNameメソッド

指定した名前(name属性の値)を持つ要素をすべて取り出し、配列として返します。

HTML5ではCSSセレクタ形式での要素ノードの取得も可能となりました。「CSSセレクタ形式で要素を取得したい」(p.412)を参照ください。

文例

myObj = document.getElementById("_header_");

id名がheaderの要素ノードを取得し、変数myObjに代入します。

myLists = document.getElementsByTagName("_li_").length;

ドキュメントに含まれるすべてのli 要素の数を取得して、変数myListsに代入します。

nameObj = document.getElementsByName("_like_").length;

ドキュメントに含まれる、name属性の値がlikeの要素の数を取得して、変数nameObjに代入します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

【SAMPLE】オブジェクトの情報を取得する ‥ P.320

DOM.02

ノードを参照したい

★.childNodes[参照番号]	特定の子ノードを参照
★.firstChild	最初の子ノードを参照
★.lastChild	最後の子ノードを参照
★.parentNode	親ノードを参照
★.previousSibiling	前のノードを参照
★.nextSibiling	次のノードを参照
● = ★.hasChildNodes()	子ノードの有無を取得
■ = ★.item(◆)	参照番号のノードを取得

★……ノードオブジェクト
●……true(子ノードがある)/false(ない)
■……取得されたノード

形式 プロパティ(childNodes、firstChild、lastChild、parentNode、previousSibiling、nextSibiling)
メソッド(hasChildNodes、item)

ノードを親子関係や兄弟関係(前後関係)などで参照するプロパティあるいはメソッドです。

childNodesプロパティ

childNodesはあるノードのすべての子ノードの配列です。特定の子ノードを参照するには、childNodes[参照番号]という形式で指定します。

firstChildプロパティ

あるオブジェクトの最初の子ノードを参照します。

lastChildプロパティ

あるオブジェクトの最後の子ノードを参照します。

parentNodeプロパティ

ある子ノードから見た親ノードを参照します。

previousSibilingプロパティ

ノードが複数並んでいる場合の前のノードを参照します。

nextSibilingプロパティ

ノードが複数並んでいる場合の次のノードを参照します。

hasChildNodesメソッド

あるオブジェクトの子ノードの有無を返すメソッドです。このメソッドには引数はありません。子をもっている場合にはtrue、子をもっていない場合にはfalseを返します。

itemメソッド

指定した参照番号のノードを返します。

文例

```
num = document.getElementById("parag01").childNodes.length;
```
id 名parag01 の子ノードの総数を変数num に代入します。

```
alert(document.getElementById('parag01').firstChild.nodeName);
```
id 名parag01 の最初の子ノードの名前をダイアログに表示します。

```
alert(document.getElementById('parag01').lastChild.nodeName);
```
id 名parag01 の最後の子ノードの名前をダイアログに表示します。

```
txtName = document.getElementById("txt01").parentNode.nodeName;
```
id 名txt01 の親ノードの名前を変数txtName に代入します。

```
alert(document.getElementById("txt01").previousSibling.id);
```
id 名txt01 の1 つ前のノードのid 名をダイアログに表示します。

```
alert(document.getElementById("txt01").nextSibiling.id);
```
id 名txt01 の1 つ後ろのノードのid 名をダイアログに表示します。

```
if(myObj.hasChildNodes()){
    alert("子ノードあり");
}
```
オブジェクトmyObjに子ノードがあった場合「子ノードあり」というダイアログを表示します。

```
myObj = document.getElementsByTagName("p").item(0);
```
p要素の最初のノードを取得し、変数myObjに代入します。

参照
ノードの種類や内容を参照したい ………… P.308
【SAMPLE】ノードを参照する ………… P.322

DOM.03

新しいノードを作成したい

● = ★.createElement(◆)　　要素ノードを作成
● = ★.createTextNode(▲)　　テキストノードを作成

●……作成されたノード
★……Documentオブジェクト
◆……作成する要素名
▲……作成するテキスト(文字列)

形式 メソッド

ノードを新たに作成するメソッドです。作成したノードはappendChildメソッドやinsertBeforeメソッド(次項参照)でドキュメント構成内に追加する必要があります。

createElementメソッド

作成したい要素名を引数として渡すことで、新たに要素ノードを作成するメソッドです。

createTextNodeメソッド

作成したいテキスト(文字列)を引数として渡すことで、新たにテキストノードを作成します。

文例

```
newDiv = document.createElement("div");
```
新たにdiv要素を作成します。

```
newTxt = document.createTextNode("Hello!");
```
テキストノードHello!を作成します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

参照
ノードを追加したい……P.306
子ノードを削除 置換したい……P.305
【SAMPLE】エレメントを作成する……P.324

子ノードを削除／置換したい

■ = ★.removeChild(◆)　　子ノードを削除
▼ = ★.replaceChild(▲,●)　　子ノードを置換

■……削除された子ノード
★……ノードオブジェクト
◆……削除する子ノード
▼……置換された子ノード
▲……置換する子ノード
●……置換される子ノード

形式　メソッド

removeChildメソッド

指定した直接の子ノードを削除するメソッドです。

replaceChildメソッド

指定した直接の子ノードを新しい子ノードに置換します。置換対象のノード，置換する新しいノードの順番で引数を指定します。

文例

```
chap = document.getElementById("chap");
intro = document.getElementById("intro");
chap.removeChild(intro);
```

id名chapのオブジェクトからid名introの子ノードを削除します。

```
newText = document.createTextNode("beginning")
chap = document.getElementById("chap");
oldText = document.getElementById("intro");
chap.replaceChild(newText, oldText );
```

テキストノードnewTextを作成し、id名chapのオブジェクトの子ノードoldTextをnewTextに置き換えます。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

新しいノードを作成したい・・・・・・・・・・・・・・P.304
ノードを追加したい・・・・・・・・・・・・・・・・・・P.306
【SAMPLE】子ノードを削除／置換する・・・・・・P.326

DOM.05

ノードを追加したい

▼ = ★.appendChild(◆)　子ノードの末尾に追加
▼ = ★.insertBefore(◆, ▲)　特定の位置に追加
● = ★.cloneNode(■)　ノードを複製

▼……追加されたノード
★……ノードオブジェクト
◆……追加するノード
▲……基準とするノード
●……複製されたノード
■……true(子ノードも複製する)／false(複製しない)

形 メソッド

オブジェクトに子ノードを追加したり、ノードを複製するメソッドです。

appendChildメソッド

子ノードを追加するメソッドです。追加したノード◆は、指定したオブジェクト★の最後の子ノードとして追加します。appendChildメソッドは特定のオブジェクトを移動させる場合にも利用できます。

insertBeforeメソッド

特定の位置に子ノードを追加するメソッドです。1番目の引数には追加するノードを、2番目の引数にはどのノードの直前に追加するのかを指定します。

cloneNodeメソッド

ノードを複製するメソッドです。引数では子ノードもいっしょに複製するかどうかを指定します。複製する場合はtrue、しない場合はfalseを指定します。

文例

```
div = document.createElement("div");
document.body.appendChild(div);
```

div要素を作成し、ドキュメントの最後に追加します。

```
img = document.createElement("img");
document.body.insertBefore(img,firstChild);
```

イメージ要素を作成し、ドキュメントの最初の子ノードの前に追加します。

```
template = document.getElementById("template");
copyNode = template.cloneNode(true);
```

id名templateの要素をその子ノードごと複製します。

▶ ブラウザ対応表	IE10	IE9	[illegible]	Fx	Chrome	Safari	[illegible]	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照

新しいノードを作成したい ・・・・・・・・・・・・・・・ P.304
子ノードを削除、置換したい ・・・・・・・・・・・・・ P.305
【SAMPLE】エレメントを作成する ・・・・・・・・・ P.324

DOM.06

ノードの種類や内容を参照したい

★.nodeName	ノード名を参照
★.nodeType	ノードの種類を参照
★.tagName	要素名を参照
★.nodeValue	ノードの値を参照／設定
★.innerHTML	要素のHTML/XHTMLタグと内容を参照／設定
★.textContent	ノードの文字列を参照／設定
★.innerText	ノードの文字列を参照／設定
★.id	要素のidを参照／設定

★……ノードオブジェクト

形式 プロパティ

ノードの種類やノードの内容を参照または設定するプロパティです。

nodeNameプロパティ

ノード名を返します。

nodeTypeプロパティ

ノードの種類を次のような数値で返します。

ノード値	ノードの種類
1	要素
2	[illegible]
3	テキスト
4	CDATAセクション
5	実体参照
6	実体宣言
7	処理命令
8	コメント
9	ドキュメント
10	文書型宣言
11	ドキュメントフラグメント
12	記法宣言

tagNameプロパティ

要素ノードの要素名を返します。

nodeValueプロパティ

ノードの値を参照／設定します。

innerHTMLプロパティ

要素が含むHTML / XHTMLタグや要素内容をテキストとして参照／設定します。タグを含めない文字列のみの場合は、innerTextやtextContentを使用します。

textContent、innerTextプロパティ

ノードの内容（文字列）を参照／設定します。innerTextプロパティはIEの拡張でFirefoxでは使用できません。標準のtextContentプロパティはIE9以降と主要ブラウザで使用できます。

idプロパティ

要素のid名を参照／設定します。

文例

```
nName = document.getElementById("pickup").nodeName;
```

id名pickupのノードの名前を変数nNameに代入します。

```
document.write(document.getElementById("pickup").nodeType);
```

id名pickupのノードの種類を表示します。

```
if(sample1.tagName == "p") {
    alert("sample1はp要素です");
}
```

オブジェクト名sample1の要素名がpだったら「sample1はp要素です」というダイアログを表示します。

```
nValue = document.getElementById("pickup").nodeValue;
```

id 名pickup のノードの値を変数nValue に代入します。

```
document.getElementById("pickup").innerHTML = "<a href='http://www.ank.co.jp/'>株式会社アンク</a>"
```

id名pickupの内容を株式会社アンクへのリンクに設定します。

```
document.getElementById("pickup").textContent = "株式会社アンク"
```

id名pickupの内容を株式会社アンクに設定します。

```
alert(document.getElementByTagName("p")[0].id);
```

ドキュメント内の最初のp要素のid名をダイアログに表示します。

▶ [illegible]対応表	IE10	IE9	IE8	Fx	Chrome	Safari	[illegible]	iOS6	Android
innerText	○	○	○	×	○	○	○	○	○
textContent	○	×	×	○	○	○	○	○	○
その他	○	○	○	○	○	○	○	○	○

参照 ノードを参照したい ・・・・・・・・・・・・・・・・・・ P.302

DOM.07

属性を参照したい

★.attributes	属性を参照
▲ = ★.getAttribute(◆)	属性の値を取得
● = ★.getAttributeNode(◆)	属性ノードを取得
■ = ★.hasAttribute(◆)	指定した属性の有無を調べる
■ = ★.hasAttributes()	属性の有無を調べる

★……ノードオブジェクト
▲……属性値
◆……属性名
●……属性ノード
■……trueまたはfalse(属性がある：true／ない：false)

形式 プロパティ(attributes)
メソッド(getAttribute、getAttributeNode、hasAttribute、hasAttributes)

要素ノードの属性を操作します。

attributesプロパティ

属性のリストを参照します。

getAttributeメソッド

指定した名前の属性◆の値を取得するメソッドです。指定した属性がなかった場合は空の文字列("")を返します。

getAttributeNodeメソッド

指定した名前の属性ノードを取得するメソッドです。指定した属性ノードがなかった場合はnullを返します。

hasAttributeメソッド

◆で指定した属性の有無を調べ、ある場合にはtrue、無い場合にはfalseを返すメソッドです。

hasAttributesメソッド

属性の有無を調べ、ある場合にはtrue、無い場合にはfalseを返すメソッドです。

文例

```
atts = document.getElementsByTagName("table")[0].attributes.length;
```
最初のtable要素が持つ属性の数を変数attsに代入します。

```
img1 = document.getElementById("img1");
width = img1.getAttribute("width");
alert(width);
```
id名img1の要素が持つwidth属性の値をダイアログに表示します。

```
p1 = document.getElementById("p1");
idAttr = p1.getAttributeNode("id");
```
id名p1の要素でノード名がidである属性ノードを変数idAttrに代入します。

```
img1 = document.getElementById("img1");
title = img1.hasAttribute("title");
alert(title);
```
id 名img1の要素がtitle属性を持っているかどうかを調べ、その結果をダイアログに表示します。

```
div1 = document.getElementById("div1");
hasAttr = div1.hasAttributes();
```
id名div1の要素が属性を持っているかどうかを調べ、その結果を変数hasAttrに代入します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照

属性を作成／設定したい …………………… P.312
属性を削除したい …………………… P.314

DOM.08

属性を作成／設定したい

● = document.createAttribute(◆)　属性を作成
★.setAttribute(◆,▲)　属性と値を追加
★.setAttributeNode(●)　属性ノードを追加

●……属性ノード
◆……属性名
★……ノードオブジェクト
▲……値

形式　メソッド

要素に対して、新たに属性を作成したり、既存の属性に変更を加えるメソッドです。

createAttributeメソッド

指定された名前の属性オブジェクトを新たに作成します。作成した属性はsetAttributeNodeメソッドで要素に追加する必要があります。

setAttributeメソッド

指定された名前の値を持つ属性を新たに追加します。または指定された名前を持つ既存の属性の値を変更します。

setAttributeNodeメソッド

新たに属性ノードを追加します。

文例

```
bgCol = document.createAttribute("bgcolor");
bgCol.nodeValue = "#000000";
document.body.setAttributeNode(bgCol);
```

値を黒(#000000)に設定したbgcolor属性を作成し、bgColという名前の属性ノードとして追加します。

```
imgObj = document.createElement("img");
imgObj.setAttribute("src","accessmap.jpg");
```

オブジェクト名imgObjのイメージ要素を作成し、src属性にaccessmap.jpgを設定します。

▶ ブラウザ対応表	IE10	IE9	[illegible]	Fx	Chrome	Safari	[illegible]	[illegible]	Android
	○	○	○	○	○	○	○	○	○

参照

属性を参照したい・・・・・・・・・・・・・・・・・・・・P.310
属性を削除したい・・・・・・・・・・・・・・・・・・・・P.314

DOM.09

属性を削除したい

● = ★.removeAttribute(◆)　属性を削除
■ = ★.removeAttributeNode(▲)　属性ノードを削除

●……削除された属性
★……ノードオブジェクト
◆……属性名
■……削除された属性ノード
▲……属性ノード

形　メソッド

要素から属性や属性ノードを削除するメソッドです。

removeAttributeメソッド

指定した属性◆を削除します。

removeAttributeNodeメソッド

指定した名前の属性ノード▲を削除するメソッドです。

文例

```
document.getElementById("div1").removeAttribute("align");
```

id名div1の要素からalign属性を削除します。

```
p1 = document.getElementById("p1");
p1.removeAttributeNode(attr);
```

id名p1の要素からatrrという名前の属性ノードを削除します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照
属性を参照したい……P.310
属性を作成、設定したい……P.312

DOM.10

CSSのスタイルを操作したい

★.style.◆

★……オブジェクト
◆……スタイルプロパティ

形式 プロパティ

HTML要素のCSSスタイルを操作するには、要素のstyleプロパティに格納される、スタイル宣言オブジェクトを利用します。このオブジェクトには、CSSプロパティにアクセスするためのプロパティが、1対1対応で用意されています。プロパティ名は以下の規則で対応しています。

CSSプロパティ	スタイルプロパティ	説明
width	width	「-」（ハイフン）が含まれていないプロパティはCSSのプロパティと同じ
font-weight border-top-style	fontWeight borderTopStyle	「-」（ハイフン）が含まれているプロパティでは「-」を除き、「-」の次の文字を大文字にしてつなぐ

document.fontColor()やdocument.bgColor()などの、文字装飾等、直接外観を操作するメソッドは、本項や次項で解説するようなCSS操作によって原則として代替可能です。

文例

```
myElement.style.fontSize = "smaller";
```

要素のフォントサイズをsmallerに設定します。

ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照 スタイルシートを操作したい……………P.316

DOM.11

スタイルシートを操作したい

★ = document.styleSheets[参照番号]	スタイルシートの取得
◆ = ★.rules[参照番号] / ★.cssRules[参照番号]	CSSルールの取得
★.insertRule(▲, 挿入位置)	CSSルールの挿入
★.addRule(●, ■)	CSSルールの追加
★.deleteRule(参照番号)	CSSルールの削除

★……スタイルシート・オブジェクト
◆……CSSルール・オブジェクト
▲……CSSルール文
●……CSSセレクター文
■……スタイル宣言文

形式 プロパティ(styleSheets、rules、cssRules)
メソッド(insertRule、addRule、deleteRule)

個別の要素のCSSスタイルだけではなく、CSSファイルやstyle要素に記述するスタイル設定も、オブジェクトとして参照・操作できます。

documentオブジェクトのstyleSheetsプロパティには、ページ内のstyle要素や外部CSSファイルを表すスタイルシート・オブジェクトが、配列形式で格納されます。

スタイルシート・オブジェクトのrulesプロパティ(IE8までのIE)またはcssRulesプロパティ(IE9以上と主要ブラウザ)には、同様に、CSSルール・オブジェクトが配列形式で格納されます。このCSSルール・オブジェクトは、「div#id{color:black}」のようなCSS指定の単位となる文(CSSルール)を表します。

CSSルールを特定のスタイルシートに追加するには、insertRule()メソッド(IE9以上と主要ブラウザ)かaddRule()メソッド(IE8までのIE)を使用します。

insertRule()メソッドでは第1引数にCSSルールの記述を文字列で指定します。次の引数に、挿入する位置を数値で指定します。CSSでは後の位置にあるルールが優先されるため、上書きしたい場合は通常、最後(★.rules.length)に挿入します。

addRule()メソッドでは、第1引数にCSSセレクターを、第2引数にスタイル宣言を指定します。CSSセレクターはCSSルールの「{}」の前にある対象指定部分で、スタイル宣言は「{}」の中にある内容指定部分です。必ず最後の位置に挿入されます。

insertRule()メソッドでは、「@media」などの@ルールも挿入できます。

たとえば、外部CSSのインポートを「@import」CSSルールによって行うことができます。「@import」CSSルールはスタイルシートの冒頭にある必要があるので、「insertRule('@import url("out.css")', 0)」のようになります。

CSSルールを削除するには、deleteRule()メソッドに削除したいCSSルールの参照番号を指定します。

文例

```
var sheet = document.styleSheets[0];
```
1番目のスタイルシートを取得します。

```
var rule = sheet.cssRules[0];
```
その1番目のCSSルールを取得します。

```
sheet.insertRule("body { color: green; }", sheet.cssRules.length);
```
スタイルシートの最後にCSSルールを追加します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照

CSSのスタイルを操作したい・・・・・・・・・・・・・・P.315
スタイルシートのCSSルールを操作したい・・・P.318
【SAMPLE】スタイルシートを操作する・・・・・・P.328

DOM.12

スタイルシートのCSSルールを操作したい

★.type	CSSルールのタイプを表す[illegible]値
★.cssText	そのルールのCSS記述文
◆.selectorText	CSSセレクター
◆.style	スタイル宣言オブジェクト
▲.encoding	文字コード名
●.media	インポートしたスタイルシートの対象メディアタイプ
●.href	インポートしたスタイルシートのURL
●.styleSheet	インポートしたスタイルシートのオブジェクト
■.media	@mediaルールの対象メディアタイプ
■.cssRules	@mediaブロックに属するCSSルールのリスト

★……CSSルール・オブジェクト(共通)
◆……CSSルール・オブジェクト(スタイル・ルール)
▲……CSSルール・オブジェクト(@charsetルール)
●……CSSルール・オブジェクト(@importルール)
■……CSSルール・オブジェクト(@mediaルール)

形式 プロパティ

スタイルシート・オブジェクト経由で取得できるCSSルール・オブジェクトは、それが表しているCSSルールの種類によって持っているプロパティが異なります。

種別は共通のtypeプロパティに格納されます。「1」がスタイル・ルール(@ルール以外の通常の指定)、「2」が@charset、「3」が@import、「4」が@media、「5」が@font-face、「6」が@page、「0」が未サポートのルールです。サンプルに記載の定数も参照ください。

スタイル・ルールのselectorTextプロパティには、CSSルールの記述の「{}」の[illegible]の対象指定(CSSセレクター)が、styleプロパティにはスタイル宣言オブジェクトが格納されます。

スタイル宣言オブジェクトには、CSSプロパティと対応した各プロパティがあり、スタイルの取得・設定が可能です(詳細は「CSSのスタイルを操作したい」p.315を参照ください)。

@charsetルールには、encodingプロパティが存在し、文字コード指定を取得できます。通常、CSSの読み込みは完了しているため、変更しても反映はされません。

@importルールにはhrefプロパティ、mediaプロパティ、styleSheetプロパティが存在し、styleSheetプロパティの中身はインポートされたスタイルシートのオブジェクトです。

@mediaルールにはmediaプロパティに加え、cssRulesプロパティとinsertRule()メソッド、deleteRule()メソッドが存在し、@mediaブロック内のCSSルールにアクセスできます。

また、共通のcssTextプロパティで、そのCSSルールの記述内容を文字列で取得できます。

文例

```
var rule = document.styleSheets[0].cssRules[0];
```

cssルールを取得します。

```
document.wrilte(rule.cssText);
```

取得したCSSルールの記述を出力します。

▶ ブラウザ対応表	IE10	IE9	[illegible]	Fx	Chrome	[illegible]	Opera	[illegible]	Android
	○	○	○	○	○	○	○	○	○

参照

CSS のスタイルを操作したい・・・・・・・・・・・・・P.315
スタイルシートを操作したい・・・・・・・・・・・・・P.316
【SAMPLE】スタイルシートを操作する・・・・・・P.328

オブジェクトの情報を取得する

入力テキストが未入力のときにエラーメッセージを表示するサンプルです。[OK]ボタンがクリックされるとclickOK関数を呼び出します。すべてのテキスト入力フィールドの入力チェックを行い、未入力の場合<span>タグ内にエラーメッセージを表示します。

loadPage関数は、XHTMLの<input>タグにonclick属性を設定したときと同じことをしています。getElementByIdメソッドは、その要素の読み込み終了時でないとエラーになるため、1行目でonloadイベントを設定し、ページ読み込み終了時にこの関数を呼び出しています。

JavaScript

```
window.onload = loadPage; //イベント[illegible]

//ページ読み込み時の処理をする関数
function loadPage(){
   document.getElementById("ok").onclick = clickOK;
}

//選択した文字列をテキストボックスに表示させる
function clickOK(){
   for(var i = 0; i < document.getElementsByTagName("p").length; i++){
      var textID = "text" + i;
      var spanID = "span" + i;
      if(document.getElementById(textID).value == ""){
         document.getElementById(spanID).innerHTML = "※この項目は必須です！
<br />";
      }else{
         document.getElementById(spanID).innerHTML = "";
      }
   }
}
```

HTML ※文字色は外部CSSで指定しています

```
<body>
  <div>
  <p>
    お名前：<input type="text" id="text0" />
    <span id="span0"></span>
  </p>
  <p>
    フリガナ：<input type="text" id="text1" />
    <span id="span1"></span>
  </p>
  <p>
    住所：<input type="text" id="text2" />
    <span id="span2"></span>
  </p>
  <p>
    電話番号：<input type="text" id="text3" />
    <span id="span3"></span>
  </p>
  <p>
    メールアドレス：<input type="text" id="text4" />
    <span id="span4"></span>
  </p>
  <input type="button" id="ok" value="OK" />
  </div>
</body>
```

Internet Explorer

未入力欄に対してメッセージが表示されます

参照

getElementById メソッド ・・・・・・・・・・・・・・ P.300
getElementsByTagName メソッド ・・・・・・・ P.300

DOM.SAMPLE-02

ノードを参照する

ノード情報を参照して表示するサンプルです。このサンプルではhead要素の情報を参照しています。参照するノードが存在する場合はobjectを返し、そのノードの情報をさらに参照できます。ノードが存在しない場合はnullを返します。

JavaScrip

```
window.onload = loadPage; //ページロード時のイベントを設定

//ページロード時の処理
function loadPage(){
var html = "";
   var div1 = document.getElementById("div1");
   var head = document.getElementById("head");
   //ノード情報を参照してHTMLを作成
   html += "親ノード(parentNode):" + head.parentNode.nodeName + "<br />";
   html += "1つ前の兄弟ノード(previousSibling):" + head.previousSibling + "<br
/>";
   html += "1つ後の兄弟ノード(nextSibling):" + head.nextSibling.nodeName + "<br
/>";
   html += "子ノードがあるか(hasChildNodes):" + head.hasChildNodes() + "<br
/>";
   html += "子ノードの数(childNodes):" + head.childNodes.length + " (";
   //すべての子ノードのnodeNameを取得
   for(i=0;i<head.childNodes.length;i++){
      html += head.childNodes.item(i).nodeName+" ";
   }
   html += ")<br />";
   //ノード情報を参照してHTMLを作成
   html += "最初の子ノード(firstChild):" + head.firstChild.nodeName + "<br />";
   html += "最後の子ノード(lastChild):" + head.lastChild.nodeName + "<br />";
   //作成したHTMLをdivタグ内に表示させる
   div1.innerHTML = html;
}
```

HTML

```
<body>
    <div id="div0">
        <b>head要素の情報</b>
    </div>
    <hr />
    <div id="div1">
    </div>
</body>
```

Internet Explorer

参照

childNodes プロパティ ……………… P.302
firstChild プロパティ ……………… P.302
lastChild プロパティ ……………… P.302
parentNode プロパティ ……………… P.302
previousSibiling プロパティ ……………… P.302
nextSibiling プロパティ ……………… P.302
hasChildNodes メソッド ……………… P.302
item メソッド ……………… P.302

DOM.SAMPLE-03

エレメントを作成する

JavaScriptで<option>タグを作成するサンプルです。ページロード時にあらかじめ作成した配列の要素を利用して、option要素を作成します。document.createElement("option")でoption要素を作成し、そのvalue値に配列の要素(URI)を指定しています。ただし「選択して下さい」ではvalue属性にdefalutという文字列を代入しておき、この項目では何も処理を行わないようchangeSelect関数で分岐させるようにしています。

また、画面表示用のテキストはcreateTextNodeメソッドで作成し、appendChildメソッドでoption要素に追加しています。最後にselect要素のidを指定してappendChildメソッドでoption要素を追加します。

JavaScript

```
var txts;
var values;

//ページロード時にoption属性を生成する関数
function pageLoad() {
    selectElem = document.getElementById("select");
    txts = new Array("選択して下さい", "株式会社アンク", "株式会社翔泳社");
    values = new Array("default", "http://www.ank.co.jp/", "http://www.shoeisha.
co.jp/");
    //配列の要素分だけoption要素を生成
    for(i=0;i<txts.length;i++){
        var elem = document.createElement("option"); //option要素を生成
        var str = document.createTextNode(txts[i]); //配列の要素でテキストを生成
        elem.value = values[i]; //option要素のvalue値に配列の要素を代入
        elem.appendChild(str); //生成したoption要素にテキストを追加
        selectElem.appendChild(elem);//option要素を追加
    }
}

//選択された項目のWebページを開く関数
function changeSelect(){
    var arrayValue = values[selectElem.selectedIndex];
    if(arrayValue == "default"){
        return;
    }
    window.open(arrayValue, "", "");
}
```

HTML

```
<body onload="pageLoad()">
    <p>
        <b>option属性をJavaScriptで生成しています</b>
    </p>
    <p>
        <select id="select" onchange="changeSelect()">
        </select>
    </p>
</body>
```

Internet Explorer

JavaScriptでプルダウンメニューを作成します

参照

createElement メソッド ・・・・・・・・・・・・・・ P.304
createTextNode メソッド ・・・・・・・・・・・・・ P.304
appendChild メソッド ・・・・・・・・・・・・・・ P.306

DOM.SAMPLE-04

子ノードを削除／置換する

質問テキストがクリックされた際に回答を表示するサンプルです。質問テキストがクリックされるとclickQ関数を呼び出し、appendChildメソッドで回答を表示するノードを作成し、表示します。すでに他の回答を表示している場合はremoveChildメソッドで削除します。

JavaScript

```
window.onload = loadPage; //イベントの設定

//変数の設定
var num = "";
var answers = new Array("A1：", "A2：", "A3：", "A4：", "A5：");

//ページ読み込み時の処理をする関数
function loadPage(){
    //<h4>タグのid="q0~4"までのノードにonclickイベントを設定
    for(var i=0;i<5;i++){
        document.getElementById("q"+i).onclick = function(){
            clickQ(this.id)
        }
    }
}

//id="q0~4"までのノードのonclickイベントに設定した関数
function clickQ(elemID){
    if(num != ""){ //他の回答が表示されている場合
        var reDiv = document.getElementById("div1"+num);
        document.getElementById("div"+num).removeChild(reDiv); //ノードを削除
    }
    //引数で取得したノードのid(q0~q4)の数字部分を抜き出して変数numに代入
    num = elemID.charAt(1);
    var elemDiv = document.getElementById("div"+num); //クリックされたノード
    var newElem = document.createElement("div"); //新しいエレメントを作成
    newElem.id = "div1"+num; //idをdiv1+num(div10~14)に設定
    newElem.innerHTML = "<span>"+answers[num]+"</span>";
                                                    //ノードの中にHTML(回答)を挿入
    elemDiv.appendChild(newElem); //クリックされたノードの最後尾に作成したノードを挿入
}
```

HTML

※文字色は外部CSSで指定しています

```
<body>
  <div id="div0">
    <p id="q0">Q1：</p>
  </div>
  <div id="div1">
    <p id="q1">Q2：</p>
  </div>
  <div id="div2">
    <p id="q2">Q3：</p>
  </div>
  <div id="div3">
    <p id="q3">Q4：</p>
  </div>
  <div id="div4">
    <p id="q4">Q5：</p>
  </div>
</body>
```

Internet Explorer

質問テキストをクリックすると、回答が表示されます

removeChild メソッド ・・・・・・・・・・・・・・・・ P.305

DOM.SAMPLE-05

スタイルシートを操作する

取得したスタイルシート・オブジェクトにCSSルールを追加し、結果を画面に出力しています。画面の文字の大きさは、最初に追加したルールによるものが表示に反映されています。最初の部分の画面出力用のオブジェクト定義の部分は、独自オブジェクトを定義して、バッファを利用した画面出力を行えるようにしています。

writeCSSRules関数は@importの場合は、インポートしたCSSに自分自身を入れ子になる形で再度適用しています。種別の判定には以下のCSSルール・オブジェクトの定数を使用します。

定数名	値	定数名	値
UNKNOWN_RULE	0	MEDIA_RULE	4
STYLE_RULE	1	FONT_FACE_RULE	5
CHARSET_RULE	2	PAGE_RULE	6
IMPORT_RULE	3		

このサンプルはIE9以上と標準準拠の他のブラウザに対応しています。

外部CSS

```
body{
    background-color: black;
    color:green;
}
```

CSS

```
@import url(dom05.css) screen;

@media print{
    body{
        background-color:white;
        color:black;
    }
}

@media screen{
    hr{width:50%;}
}
```

JavaScript

```
////////////////////// ここから画面出力用オブジェクト定義
function Output() { // コンストラクター定義
    this.buffer = ""; // バッファの初期化
}
Output.prototype.add = function(text) { // addメソッド定義
    this.buffer += text; // バッファに追加
};
Output.prototype.write = function() { // writeメソッド定義
    if (this.el === undefined) { // 初回使用時に初期化
        this.el = document.getElementById("output");
    }
    this.el.innerHTML = this.buffer; // バッファを出力
    this.buffer = ""; // バッファのクリア
};
////////////////////// ここまで画面出力用オブジェクト定義

// 画面出力用オブジェクトの生成
var output = new Output();

// CSS書き出し処理
function writeCSSRules(sheet) {

    if (!sheet) {
        return;
    }
    // すべてのCSSルールについて処理
    output.add("<ol>"); // 出力用オブジェクトに書き込み
    for (var i = 0; i < sheet.cssRules.length; i++) {
        var rule = sheet.cssRules[i];
        // 取得したルールのCSS記述を出力
        output.add("<li>" + rule.cssText);
        // @importルールか判定
        if (rule.type === rule.IMPORT_RULE) {
            // インポートしたCSSについて再帰的に処理
            writeCSSRules(rule.styleSheet);
        }
        output.add("</li>");
    }
    output.add("</ol>");
}

// 初期化処理
window.onload = function() {
    var sheet = document.styleSheets[0];
    // CSSルールの追加
```

```
    sheet.insertRule("div#output {font-size: 20pt;}", 1);
    writeCSSRules(sheet); // CSSの内容を書き出し
    output.write(); // オブジェクトに蓄積した内容を出力
};
```

HTML

```
<body>
  <div id="output"></div>
</body>
```

Internet Explorer

動的に追加した結果のCSSが表示されます

参照
cssRules プロパティ ･････････････････ P.316
cssText プロパティ ･････････････････ P.318

ASYNC.01

非同期通信を利用したい

★ = new XMLHttpRequest()　XMLHttpRequesオブジェクトを作成

★……XMLHttpRequestオブジェクト

形式 オブジェクト

XMLHttpRequestはHTTPプロトコルを使ってサーバーと通信し、テキスト形式やXML形式のデータを送受信するオブジェクトです。サーバーと非同期に通信を行えることから、Ajax(ページを遷移することなく、動的にページの一部を書き換えることが可能な技術)を実現するための技術の1つになっています。

なお、Internet Explorer 7より前のバージョンではXMLHttpRequestオブジェクトに対応していません。代わりにActiveXでXMLHttpRequestと同等の機能(XMLHTTPオブジェクト)が提供されており、次のような書式でオブジェクトを生成します。

IE6の場合　★ = ActiveXObject("Msxml2.XMLHTTP")
(★……XMLHTTPオブジェクト)

生成するオブジェクトの種類を判断するには、まずnewステートメント(p.039)を使ってXMLHttpRequestオブジェクトを生成してみて、正しく生成できるかどうかで判断する手法がよく用いられます。

文例

```
xmlhttp = new XMLHttpRequest();
```
XMLHttpRequest オブジェクトを作成します。

```
xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
```
XMLHTTP オブジェクトを生成します(IE6の場合)。

▶ ブラウザ対応表	IE10	[illegible]	IE8	[illegible]	Chrome	[illegible]	[illegible]	[illegible]	Android
	○	○	○	○	○	○	○	○	○

【SAMPLE】非同期通信と同期通信 ･････････ P.341

ASYNC.02

サーバーへのリクエストを送信したい

リクエストを初期化

リクエストを送信

★……XMLHttpRequestオブジェクト
◆……HTTPメソッド名(POST、GETなど)
▲……リクエスト先URI
●……trueまたはfalse（非同期:true／同期:false）【省略可】
■……ユーザー名【省略可】
▼……パスワード【省略可】
☆……送信データ

形式 メソッド

サーバーへのリクエスト送信時の設定とリクエストの送信を行うメソッドです。

openメソッド

HTTPリクエストを初期化し、データの送信先や送信方法を設定します。HTTPメソッド◆にはGET、POST、PUT、HEADなどがありますが、通常はGETかPOSTを指定します。リクエスト先URI▲にはリクエストの送信先あるいは読み込みたいファイル名を絶対URIまたは相対URIで指定します。なお、以上の引数は必須です。

同期通信か非同期通信かはtrue(非同期：デフォルト)またはfalse(同期)で指定します。ユーザー名とパスワードは、認証に使用されるユーザー名やパスワードの文字列です。

このメソッドを設定したあとにsendメソッドでリクエストをサーバーに送信します。

sendメソッド

サーバーにリクエストを送信します。openメソッドのHTTPメソッドがPOSTの場合は引数に送信するデータを指定できますが、GETの場合は引数にnullを指定します。

文例

```
xmlhttp.open("GET","/cgi-bin/dictionary.cgi?");
xmlhttp.send(null);
```

GETメソッドを利用した非同期のHTTPリクエストを設定し、送信します。

▶ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

参照

データを受信したい……P.335
通信を中止したい……P.336
【SAMPLE】非同期通信と同期通信……P.341

ASYNC.03

データを受信したい

★.responseText テキストデータとして取得したレスポンスを参照

★.responseXML XMLオブジェクトとして取得したレスポンスを参照

★……XMLHttpRequestオブジェクト

形式 プロパティ

HTTPリクエストに対するレスポンス本体を表すプロパティです。

responseTextプロパティ

HTTPリクエストに対するレスポンスをテキスト形式で取得します

esponseXMLプロパティ

HTTPリクエストに対するレスポンスをXML形式で取得します。取得したXMLデータは、DOMのプロパティやメソッドを使って処理できます。

文例

```
alert(xmlHttp.responseText);
```

非同期のHTTPリクエストに対し、テキスト形式で取得したデータをダイアログに表示します。

```
xmlData = xmlHttp.responseXML;
```

非同期のHTTPリクエストに対し、XML形式で取得したデータを変数xmlDataに代入します。

▶ ブラ[illegible]対応表	vIE10	IE9	[illegible]	Fx	[illegible]	Safari	Opera	[illegible]	Android
	○	○	○	○	○	○	○	○	○

参照
サーバーへのリクエストを送信したい …… P.334
通信を中止したい …… P.336

ASYNC.04

通信を中止したい

★.abort()

★……XMLHttpRequestオブジェクト

形式 メソッド

実行中の非同期通信を中止するメソッドです。

文例

```
xmlHttp.abort();
```

実行中の非同期通信を中止します。

▶ ブラウザ対応表	IE10	IE9	[illegible]	Fx	Chrome	Safari	Opera	[illegible]	Android
	○	○	○	○	○	○	○	○	○

参照
サーバーへのリクエストを送信したい ……… P.334
データを受信したい ……………………… P.335

ASYNC.05

通信の状態を調べたい

★.readyState　現在の通信状態を参照
★.status　ステータスコードを参照
★.statusText　ステータステキストを参照

★……XMLHttpRequestオブジェクト

形式 プロパティ

サーバーとの通信の状態をチェックするプロパティです。

readyStateプロパティ

現在のリクエスト処理の状態を示します。戻り値は下記の整数のいずれかになります。

0	uninitialized（未初期化）	open()がまだ呼び出されていない
1	loading（読み込み中）	open()は呼ばれたがsend()がまだ呼び出されていない
2	loaded（読み込み済み）	send()は呼ばれたが、まだステータスやヘッダがない
3	interactive（処理中）	データの一部を受け取った
4	complete（完了）	すべてのデータの読み込みが完了した

statusプロパティ　statusTextプロパティ

statusプロパティはHTTPステータスコード（リクエスト結果を示します）を、statusTextプロパティはコードに対応するステータステキストを返します。主な値は下表の通りです。

ステータスコード	ステータステキスト	意味
200	OK	リクエストに成功
401	Unauthorixed	権限がない（ユーザ認証が必要で認証に失敗したときなど）
403	Forbidden	禁止（アクセス権限がないときなど）
404	(File)Not Found	リクエストしたファイルが存在しない
500	Internal Server Error	サーバー内部にエラーが発生
503	Service Unavailable	サービス利用不可（一時的な過負荷やメンテナンスなど）

文例

```
if((xmlhttp.readyState == 4)&&(xmlhttp.status == 200)){
    alert("OK");
}
```

すべてのデータの読み込みが正常に完了したら「OK」というダイアログを表示します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

通信状態の変化に対する処理を指定したい ‥‥ P.338
【SAMPLE】非同期通信と同期通信 ‥‥‥‥‥ P.341

ASYNC.06

通信状態の変化に対する処理を指定したい

★.onreadystatechange = ◆

★……XMLHttpRequestオブジェクト
◆……イベント発生時の処理

形式 プロパティ

サーバーとの通信の状態(readyStateプロパティ)が変化するとonreadystatechangeプロパティが呼び出されます。onreadystatechangeプロパティにはreadyStateプロパティの変化に対応する処理(イベントハンドラ)を指定します。

文例

```
xmlHttp.onreadystatechange = myFunc;
```

通信の状態が変化したら関数myFuncを呼び出します。

▶ ブラウザ対応表	IE10	[illegible]	IE8	[illegible]	Chrome	[illegible]	Opera	[illegible]	[illegible]
	○	○	○	○	○	○	○	○	○

通信の状態を調べたい……………………P.337
【SAMPLE】非同期通信と同期通信…………P.341

ASYNC.07

レスポンスヘッダ情報を取得したい

● = ★.getAllResponseHeaders()　すべてのレスポンスヘッダを取得
● = ★.getResponseHeader(◆)　レスポンスヘッダを取得

●……レスポンスヘッダの情報
★……XMLHttpRequestオブジェクト
◆……取得するレスポンスヘッダ名

形式 メソッド

HTTPでサーバーから送信されるヘッダ(レスポンスヘッダ)の情報を取得するメソッドです。

getAllResponseHeadersメソッド

レスポンスヘッダのすべての情報を文字列で返します。

getResponseHeaderメソッド

レスポンスヘッダのうち、◆で指定した値を文字列で返します。

文例

```
alert(xmlhttp.getAllResponseHeader());
```

レスポンスヘッダのすべての情報をダイアログに表示します。

```
headerValue = xmlhttp.getResponseHeader("Content-Type");
```

レスポンスヘッダのContent-Typeを変数headerValueに代入します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	○	○	○	○	○	○	○

リクエストヘッダを設定したい …………… P.340

ASYNC.08

リクエストヘッダを設定したい

★.setRequestHeader(◆,▲)

★……XMLHttpRequestオブジェクト
◆……設定したいヘッダ名
▲……設定したいヘッダの値

形式 メソッド

HTTPでサーバーへ送信するヘッダ(リクエストヘッダ)を設定するメソッドです。引数で指定したヘッダ名と値をリクエストに追加します。

文例

```
xmlhttp.setRequestHeader("Content-Type","application/x-www-form-urlencoded; charset=UTF-8");
```

Content-Typeを設定します。

▶ ブラウザ対応表	IE10	[illegible]	IE8	Fx	[illegible]	Safari	[illegible]	[illegible]	Android
	○	○	○	○	○	○	○	○	○

参照 レスポンスヘッダ情報を取得した。……P.339

ASYNC.SAMPLE-01

非同期通信と同期通信

非同期通信と同期通信のサンプルです。このサンプルでは、[同期](同期通信)、または[非同期](非同期通信)ボタンがクリックされると、サーバー側で3秒間待機します。その間に通信中であることを画面上に表示し、通信が終了すると警告ダイアログを表示します。非同期通信ではサーバーとの通信中もプログラムが停止することなく他の処理を実行できるので、サーバーとの通信中に通信中であることを画面に表示しますが、同期通信では通信が終了するまで他に処理を実行できないので、表示しません。

ボタンがクリックされるとsendAsync関数を呼び出します。[同期]ボタンの場合はtrue、[非同期]ボタンの場合はfalseを引数として渡します。この関数の中でsendingMSG関数とsendRequest関数を呼び出しています。

sendingMSG関数は非同期通信中に画面に通信中の表示を出すためのタイマーをセットする関数です。

JavaScript

```
var pos = 0; // 進捗表示の位置を格納する変数
var timer = null; // タイマーを管理する変数

function sendAsync(async) { // ボタンクリックで呼び出される関数
    document.getElementById("async").disabled = true;
    sendingMSG(); //進捗を表示するタイマーの開始
    sendRequest("GET", "async.cgi", async, null, finish);
}

//リクエストを送信する処理
function sendRequest(method, uri, async, data, callback) {
    var xmlhttp; //XMLHttpRequestオブジェクトの作成
    try {
        xmlhttp = new XMLHttpRequest();
    } catch (e) {
        try {
            xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
        } catch (e) {
            return;
        }
    }
    xmlhttp.onloadend = callback; //通信が終了したときの処理を設定
    xmlhttp.open(method, uri, async); //リクエストの設定
    //キャッシュ回避のヘッダーを設定
```

```
    xmlhttp.setRequestHeader('Pragma', 'no-cache');
    xmlhttp.setRequestHeader('Cache-Control', 'no-cache');
    xmlhttp.setRequestHeader('If-Modified-Since', new Date(0).toGMTString());
    xmlhttp.send(data); //リクエストを送信
}

function finish(e) { //sendRequest関数の第5引数に渡している(callback)関数
    if (e.target.readyState === 4) { //readyStateが4(通信完了)のとき
        document.getElementById("div2").innerHTML = "";
        document.getElementById("async").disabled = false;
        clearInterval(timer);
        alert("通信完了しました。");
    }
}

function sendingMSG() { //タイマーをセットする関数
    document.getElementById("div2").innerHTML = "<p>通信中…</p><canvas
id='progress' width='100' height='20'></canvas>";
    timer = setInterval("progress()", 50); //タイマーをセット
}

function progress() { // 通信中であることを画面上に表示させる関数
    var c = document.getElementById("progress").getContext("2d");
    if (c) { //Canvasを使用してアニメーションを描画
        c.fillStyle = "white";
        c.fillRect(0, 0, c.canvas.width, c.canvas.height); //背景を塗りつぶす
        c.fillStyle = "green";
        c.fillRect(pos, 0, c.canvas.width, c.canvas.height); //緑のバーを描画
        pos = pos + (c.canvas.width / 4); //バーの位置を移動
        if (pos > c.canvas.width) { //右端を越えたら左端から
            pos = c.canvas.width * -1;
        }
    }
}
```

HTML ※レイアウトは外部CSSで指定しています

```
<body>
<form action="">
    <div id="div1">
        <b>サーバー側で3秒停止させます</b>
        <input type="button" id="sync" value="同期"
onclick="sendAsync(false)" />
        <input type="button" id="async" value="非同期"
onclick="sendAsync(true)" />
    </div>
    <hr />
    <div id="div2">
    </div>
</form>
</body>
```

CGI

```
#!C:¥tools¥perl¥bin¥perl

sleep 3;
```

ここではsendReqest関数について説明しましょう。

①5つの引数を受け取っています。

第1引数	method	送信方法(GETやPOSTなど)
第2引数	uri	送信先ファイル名
第3引数	async	非同期通信ならtrue、同期通信ならfalse
第4引数	data	送信先へ渡すデータ
第5引数	callback	通信終了時に呼び出すコールバック関数

②XMLHttpRequestオブジェクトxmlhttpを作成します。
XMLHttpRequestオブジェクトはnewステートメント(p.037)を使用して作成しますが、Internet Explorer 6以前のブラウザの場合はXMLHttpRequestではなくActiveXObjectを使用します。
Internet Explorer 6　　ActiveXObject("Msxml2.XMLHTTP")

③XMLHttpRequestオブジェクトxmlhttpのloadendイベントに引数で受け取ったコールバック関数を設定します。このイベントは成功失敗に関わらず通信が終了したときに発生します。関数の中で、通信が成功したことをreadyStateプロパティで確認しています。

④openメソッドを呼び出します。このメソッドには引数として①で受け取ったmethod、uri、asyncを渡します。IEでキャッシュされるのを防ぐためHTTPヘッダーを設定しています。

⑤sendメソッドを呼び出します。このメソッドには引数として①で受け取ったdataを渡します。このサンプルではデータの転送は必要ないのでnullを渡しています。

通信が終了するとコールバック関数に設定したfinish関数を呼び出します。この関数はsendingMSG関数でセットしたタイマーを解除、通信中の表示を削除して、ダイアログを表示します。

※同期通信の場合は通信が完了するまでプログラムが停止するので、sendingMSG関数の処理内容が実行されないままfinish関数が呼ばれます。そのため画面に何も表示しません。

[非同期]ボタンをクリックした際はサーバーとの通信中でもプログラムが停止することなく他の処理を実行できます

[同期]ボタンをクリックした際は通信が終了するまで他に処理を実行できません

通信が終了すると警告ダイアログが表示されます

参照
XMLHttpRequest オブジェクト ……… P.333
open メソッド ……… P.334
send メソッド ……… P.334
onreadystatechange プロパティ ……… P.338
readyState プロパティ ……… P.337

GRAPHICS&MEDIA.01

Canvasを利用したい

★ = ◆.getContext(▲)　　描画コンテキストの取得

★……Canvas要素への描画を担当するオブジェクト(コンテキスト)
◆……描画するCanvas要素への参照
▲……Canvasへの描画方法の指定("2d"もしくは"webgl")

形式　メソッド

HTML5のCanvas要素に描画するには、描画方法ごとに用意された、「コンテキスト」というオブジェクトを取得し、このオブジェクトのメソッドを使って描画を行います。

コンテキストには、現在のところ二次元の図形を描く「2d」とOpenGL技術を利用して3D描画を行う「webgl」が存在します。ただし、WebGLコンテキストは利用できる環境の条件が厳しく、内容も非常に複雑ですので、本書では2Dコンテキストのみ扱います。

文例

```
<canvas id="myCanvas" width="500" height="500"></canvas>
```
描画の対象となるCanvas要素をHTML内に配置します。

```
var myCanvas = document.getElementById("myCanvas");
```
Canvas要素への参照を取得します。

```
var context = myCanvas.getContext("2d");
```
2Dコンテキストを取得します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	×	○	○	○	○	○	○

参照

GRAPHICS&MEDIA.02

四角形を描画したい

★.strokeRect(◆, ▲, ●, ■)	四角形の輪郭線を描く
★.fillRect(◆, ▲, ●, ■)	四角形を描いて塗りつぶす
★.clearRect(◆, ▲, ●, ■)	四角形の領域を透明にくりぬく

★……2Dコンテキスト
◆……四角形の左上の点のX座標(ピクセル)
▲……四角形の左上の点のY座標(ピクセル)
●……四角形の幅(ピクセル)
■……四角形の高さ(ピクセル)

形式 メソッド

Canvasに四角形の線を描くにはstrokeRectメソッドを、中身を塗りつぶした四角形を描きたいときは、fillRectメソッドを使用します。

塗りつぶし色や線の太さなどの描画スタイルは、描画時点でのコンテキスト★の状態(各属性値)に従います(p.350参照)。

四角形の形に描画をクリアしたいときは、clearRectメソッドを使用します。指定した領域が透明になるため、外見上、四角形にくりぬいたようになります。

文例

```
context.fillRect(10, 10, 100, 100);
```

1辺100ピクセルの正方形を(10, 10)の位置に描きます。

Column Canvas上での座標

Canvas上での座標は、左上隅の点が(0, 0)で、右方向と下方向がプラスの方向です。たとえばCanvasの右下隅の座標は、(Canvasの幅, Canvasの高さ)になります(単位：ピクセル)。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS5	Android
	○	○	×	○	○	○	○	○	○

参照

線や塗りつぶしの色を指定したい……P.350
グラデーションを設定したい……P.351
図形を変形させたい……P.353
透明度を指定したい／影を付けたい……P.355
【SAMPLE】Canvasに描画する……P.362

GRAPHICS&MEDIA.03

パスを使って図形を描画したい

★.beginPath()	パスをクリアして新しい図形を開始
★.closePath()	終点と起点を結んでパスを閉じる
★.moveTo(X座標, Y座標)	指定の点[illegible]で移動
★.lineTo(X座標, Y座標)	直前の点から指定の点までパスを[illegible]で引く
★.fill()	パスを閉じて内部を塗りつぶして描画
★.stroke()	パスを実際に線として[illegible]

★……2Dコンテキスト

形式 メソッド

複雑な図形を描くにはパス(点をつなぐ経路)を作成します。moveTo()メソッドで指定の点へ移動し、lineTo()で指定の点まで線を引きます。

fill()メソッド(塗りつぶしあり)またはstroke()メソッド(塗りつぶしなし)で作成したパスが描画されます。fill()メソッドでは、パスは自動的に閉じられます。

作成したパスをクリアするには、beginPath()メソッドを呼びます。closePath()メソッドは、パスの終点と起点を[illegible]んで図形を[illegible]じる場合に使います。

文例

```
context.beginPath(); // パスの初期化
context.moveTo(100, 100); // 起点を設定
context.lineTo(100, 200); // 指定の点まで線を引く
context.stroke(); // 塗りつぶしなしで描画
```

(100.100)から(100.200)までの直線をパスを使って引いています。

▶ [illegible]対応表	IE10	[illegible]	IE8	Fx	[illegible]	Safari	Opera	[illegible]	[illegible]
	○	○	×	○	○	○	○	○	○

参照
線や塗りつぶしの色を指定したい……P.350
グラデーションを設定したい……P.351
図形を変形させたい……P.353
透明度を指定したい/影を付けたい……P.355
【SAMPLE】Canvasに描画する……P.362

GRAPHICS&MEDIA.04

パスを使って特定の図形を描画したい／地点がパスの中にあるかを調べたい

★.arc(X座標, Y座標, 半径, ◆, ▲ [, ●])　パスに円・円弧を追加
★.rect(X座標, Y座標, 幅, 高さ)　パスに四角形を追加
★.isPointInPath(X座標, Y座標)　点がパスの中にあるかどうかを返す

★……2Dコンテキスト
◆……円弧の開始角（単位：ラジアン）
▲……円弧の終了角（単位：ラジアン）
●……描画を反時計回りにするかの指定（省略可能。既定値はfalse）

形式　メソッド

arcメソッドを使うと、パスに円または円の一部が追加されます。最初の3つの引数で、追加する円の中心点と半径を指定します。次の引数で、開始角◆と終了角▲を指定します。

開始角◆に対応する円周上の点が円弧の起点、終了角▲の点が終点です。角度は時計でいう3時の位置を0ラジアンとして数えます。起点から終点まで、既定では時計回りに、最後の引数●がtrueなら反時計回りに、円周上を切り取った部分が、パスに追加されます。

rectメソッドを使うと、パスに四角形が追加されます。追加する四角形の左上の角の座標と、幅、高さを指定します。直前の終点からは線が引かれず、左上の角が終点になります。

点がパスが囲んでいる領域の中にあるかどうか知るには、isPointInPathメソッドを使用します。中にあればtrueを返します。このとき、変形(p.353参照)は考慮されません。

Column　度数とラジアン

ラジアンは、1ラジアンの角の作る弧の長さが半径と等しくなるように決められた単位で、360度が2πラジアンです。ラジアンへは「度数 * Math.PI / 180」の式で変換可能です。

ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	×	○	○	○	○	○	○

参照
線や塗りつぶしの色を指定したい……P.350
グラデーションを設定したい……P.351
図形を変形させたい……P.353
透明度を指定したい／影を付けたい……P.355
【SAMPLE】Canvasに描画する……P.362

GRAPHICS&MEDIA.05

線や塗りつぶしの色を指定したい

★.fillStyle = ◆	塗りつぶしのスタイルを指定
★.strokeStyle = ◆	線のスタイルを指定
★.lineWidth = ▲	線の幅を指定
★.save()	コンテキスト状態の保存
★.restore()	コンテキスト状態の復元

★……2Dコンテキスト
◆……描画スタイル(色、グラデーションまたはパターン)
▲……幅(単位:ピクセル)

形式 プロパティ(fillStyle、strokeStyle、lineWidth)
メソッド(save、restore)

描画の際の共通設定は、その時点でのコンテキストの状態(各属性値)に従います。

lineWidthプロパティでは、描画する線の幅をピクセルで指定します。

fillStyleプロパティでは、塗りつぶしのスタイルを、strokeStyleプロパティでは線のスタイルを指定します。スタイルに指定可能なのは、CSSと同様の色名や#RRGGBB形式による色指定の文字列か、グラデーションやパターンを表すオブジェクト(次項参照)です。

関数呼び出しの前後など、コンテキスト状態を一旦退避したり復元したりしたい場合には、saveメソッドで内部に状態を保存し、restoreメソッドで保存した状態を復元します。

文例

```
context.fillStyle = "red";
```

塗りつぶしの色を設定しています。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	×	○	○	○	○	○	○

参照
グラデーションを設定したい・・・・・・・・・・・・・P.351
透明度を指定したい／影を付けたい・・・・・・・・P.355
【SAMPLE】Canvasに描画する・・・・・・・・・・・P.362

GRAPHICS&MEDIA.06

グラデーションを設定したい

★.createLinearGradient(◆, ▲, ●, ■) 線形グラデーション
★.createRadialGradient(◆, ▲, ▼, ●, ■, ☆) 円形グラデーション
★.addColorStop(起点からの距離, 色指定) グラデーションの色指定

★……2Dコンテキスト
◆……直線の起点のX座標 ／ 開始円の中心のX座標
▲……直線の起点のY座標 ／ 開始円の中心のY座標
●……直線の終点のX座標 ／ 終端円の中心のX座標
■……直線の終点のY座標 ／ 終端円の中心のY座標
▼……開始円の半径
☆……終端円の半径

形式 メソッド

線形グラデーションは、createLinearGradient()メソッドで作成します。色彩は起点(◆, ▲)と終点(●, ■)を結ぶ直線に沿って変化していきます。

円形グラデーションは、createRadialGradient()メソッドで作成します。色彩は開始円の円周から終端円の円周へと放射状に変化していきます。

addColorStopメソッド

上記で作成したグラデーションのオブジェクトには、まだ各地点での色が指定されていません。オブジェクトのaddColorStop()メソッドで、各地点の色を指定します。起点からの距離は、全体に対する、0.0から1.0の割合で指定し、次にその地点の色を文字列で指定します。

文例

```
var grd = context.createLinearGradient(0, 0, 100, 100);
grd.addColorStop(0, "red");
grd.addColorStop(1, "blue");
context.fillStyle = grd;
```

赤から青へのグラデーションを作成して塗りつぶしのスタイルとして設定しています。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	×	○	○	○	○	○	○

参照

線や塗りつぶしの色を指定したい・・・・・・・・・・P.350
透明度を指定したい／影を付けたい・・・・・・・・P.355
【SAMPLE】Canvasに描画する・・・・・・・・・・・P.362

GRAPHICS&MEDIA.07

画像を表示・操作したい

★.drawImage(◆, ▲, ●[, ■, ▼])

★.drawImage(◆, ☆, ◇, △, ○, ▲, ●, ■, ▼) 画像の一部を切り出して描画

★……2Dコンテキスト
◆……画像オブジェクト
▲……画像の左上のX座標
●……画像の左上のY座標
■……画像の描画時の幅(省略可能)
▼……画像の描画時の高さ(省略可能)
☆……切り出す領域の左上のX座標(元画像内の座標)
◇……切り出す領域の左上のY座標(元画像内の座標)
△……切り出す領域の幅
○……切り出す領域の高さ

形式 メソッド

画像を描画するにはdrawImageメソッドを使用します。第1引数には、画像オブジェクト◆(IMG要素の参照か、new Image()で作成)を使用します。

画像全体を描画する場合には、画像オブジェクトの次の2つの引数に、画像が描画される位置(▲, ●)を指定します。この位置が画像の左上隅になるように描画されます。画像を縮小拡大する場合は、次の引数に、画像の幅■と高さ▼を指定します。

画像の一部を切り出して描画する場合は、画像オブジェクト◆と描画位置(▲, ●)の間に、切り出す領域の指定が割り込みます。まず、切り出す領域の左上隅の位置(☆, ◇)を元画像の左上隅を原点とする座標で指定し、次に切り出す領域の幅△と高さ○を指定します。その後、切り出した画像を描画する位置(▲, ●)と幅■、高さ▼の指定が続きます。

サンプルの記述とその描画結果も参照してください。

文例

```
context.drawImage(image, 50, 50, 100, 100, 30, 30, 200, 200)
```

画像から一部を切り取って指定の位置に描画しています。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	×	○	○	○	○	○	○

【SAMPLE】Canvasに描画する ………… P.362

GRAPHICS&MEDIA.08

図形を変形させたい

メソッド	説明
★.scale(◆, ▲)	拡大
★.rotate(●)	回転
★.translate(■, ▼)	移動
★.transform(成分a, 成分b, 成分c, 成分d, 成分e, 成分f)	行列による変換の追加
★.setTransform(成分a, 成分b, 成分c, 成分d, 成分e, 成分f)	行列による変換の指定

★……2Dコンテキスト
◆……横方向の拡大縮小率
▲……縦方向の拡大縮小率
●……回転させる角度(単位:ラジアン)
■……右方向の移動距離(単位:ピクセル)
▼……下方向の移動距離(単位:ピクセル)

メソッド

Canvasでは図形の変形は、コンテキスト・オブジェクト★が状態として持っている、「どう変換するかを決めるルール」を、描画の際にその都度適用する、という形をとります。この変換ルールのことを、「変形マトリックス」と呼びます。

変形の手順としては、変形用のメソッドを使って必要な変形をすべて変形マトリックスに追加してから、strokeメソッドなどの描画を行うメソッドを呼び出します。追加された各変形は、描画時には、追加された順番で、変換結果にさらに変換を加えていく形で適用されます。

拡大縮小の変形を加えるには、scaleメソッドを呼びます。横方向の拡大率◆、縦方向の拡大率▲を引数に指定します。回転の変形を加えるには、rotateメソッドを回転角度を引数にして呼びます。移動の変形を加えるには、translateメソッドを、右方向、下方向への移動距離を引数にして呼びます。マイナス値を指定すると反対方向の指定になります。

変形マトリックスの現在の状態や、可能なすべての変形は、数学でいう3×3の行列式で表すことができます。そこで、行列によって変形を指定することもできます。

行列を指定して、カスタムな変形を変形マトリックスに追加するには、transformメソッドを呼び出します。追加ではなく、指定した行列の状態に変形マトリックスを初期化するには

setTransformメソッドを呼び出します。指定できる行列は下記の形式のもので、その各成分a, b, c, d, e, fの数値を、この[illegible]で引数に指定します。

a, c, e
b, d, f
0, 0, 1

変形マトリックスが上記の時、点(x, y)は下記の計算に基づいて(x', y')に変換されます。

$$x' = ax + cy + e$$
$$y' = bx + dy + f$$

文例

```
context.translate(100, 200);
context.scale(2, 2);
context.stroke()
```

右に100、下に200移動し、横に2倍、縦に2倍の拡大をして描画しています。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	×	○	○	○	○	○	○

参照

四角形を描画したい・・・・・・・・・・・・・・・・・・・・P.347
パスを使って図形を描画したい・・・・・・・・・・・P.348
パスを使って特定の図形を描画したい／地点がパスの中にあるかを調べたい・・・・・・・・P.349
【SAMPLE】Canvasに描画する・・・・・・・・・・・P.362

GRAPHICS&MEDIA.09

透明度を指定したい／影を付けたい

★.globalAlpha = ◆	透明度を指定
★.shadowBlur = ●	影のぼかしを指定
★.shadowOffsetX = ■	X軸方向に影を付ける
★.shadowOffsetY = ▼	Y軸方向に影をつける
★.shadowColor = ☆	影の色を指定

★……2Dコンテキスト
◆……透明度（0.0から1.0）
●……影のぼかしレベルの数値
■……影のX軸方向の距離
▼……影のY軸方向の距離
☆……影の色指定

形式 プロパティ

コンテキスト状態の設定によっては、その他の視覚的効果を付け加えることができます。

描画の透明度を指定するには、globalAlphaプロパティに「0.0」（透明）から、「1.0」（不透明）の間の数値を指定します。既定値は「1.0」（不透明）です。

影の色はshadowColorプロパティに指定します。本体から影をどれだけずらして表示するかは、shadowOffsetXプロパティとshadowOffsetYプロパティにそれぞれX軸方向の距離、Y軸方向の距離を指定します。影を滲んだようにぼかすには、shadowBlurプロパティに0以上の数値を指定します。数値が大きいほどぼかし効果が大きくなります。

文例

```
context.globalAlpha = 0.5;
context.fill()
```

透明度0.5で描画を行っています。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	×	○	○	○	○	○	○

線や塗りつぶしの色を指定したい……P.350
グラデーションを設定したい……P.351
【SAMPLE】Canvasに描画する……P.362

GRAPHICS&MEDIA.10

文字列を表示したい

★.strokeText(◆, ▲, ● [, ■])　文字列を輪郭線で描画
★.fillText(◆, ▲, ● [, ■])　文字列を描画して塗りつぶす
▼ = ★.measureText(◆)　文字列の[illegible]を取得

★……2Dコンテキスト
◆……描画する文字列
▲……X座標(単位：ピクセル)
●……Y座標(単位：ピクセル)
■……最大幅(単位：ピクセル)
▼……幅などの[illegible]情報を持つTextMetricsオブジェクト

形式　メソッド

文字列を描画するには、strokeTextメソッドかfillTextメソッドを使用します。strokeTextメソッドでは輪郭線のみの[illegible]になります。引数は共通で、描画する文字列に続けて描画位置を座標で指定します。最後の引数は最大幅で、これをはみ出した部分は描画されません。

フォント名やサイズの指定は、コンテキスト★のfont属性にCSSのfontプロパティの指定と同様の指定をします。左右の寄せは、textAlign属性に、「left」(左寄せ)、「right」(右寄せ)、「center」(中央寄せ)を指定します。同様に上下の寄せは、textBaseLine属性に、「top」(上寄せ)、「middle」(上下の中央寄せ)、「bottom」(下寄せ)を指定します。

measureTextメソッドで文字列を引数にして呼ぶと返されるTextMetricsオブジェクト▼には、文字列の描画時の情報が格納されます。現在、width属性(文字幅)が利用可能です。

文例

```
context.font = "bold 12pt sans-serif";
context.fillText("ようこそ", 100, 100);
```

(100, 100)の位置に文字列を描画しています。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	×	○	○	○	○	○	○

参照　【SAMPLE】Canvasに描画する‥‥‥‥‥‥P.362

SVGを操作したい

★ = ◆.getSVGDocument()

SVGファイル内のSVG要素を取得

★……SVG要素
◆……SVGファイルを参照しているembedまたはobject要素

形式 メソッド

SVGはXMLの一種であるため、DOMを利用して要素や属性を操作することができます（DOMの操作については p.300参照）。

HTMLのなかに直接SVG要素が記述されている場合は、通常のDOM操作と同じく、document.getElementById()などでSVG要素への参照を取得し、必要に応じて属性値を設定したり、要素を作成して追加したりすることでSVG画像を操作します。

文例

```
var rect = svg.getElementsByTagName("rect")[0];
```

SVG要素内の、四角形を表すrect要素への参照を取得します。

ただし、SVGの要素を新規に作成するときは document.createElementNS("http://www.w3.org/2000/svg",SVGのタグ名) とする必要がある点に注意してください。document.createElement()で作成した要素はSVGの要素としては使用できません。

getSVGDocumentメソッド

objectまたはembed要素でHTMLにSVGファイルを埋め込んでいる場合には、getSVGDocument()を使ってSVGファイル内にあるSVG要素への参照を取得します。SVGファイルの読み込みが終わっている必要があるため、onloadイベント等を利用して読み込みを待つ必要があります。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	×	○	○	○	○	○	○

オブジェクトの情報を取得したい・・・・・・・・・・P.300

GRAPHICS&MEDIA.12

音声・動画の再生を操作したい

★.play()	再生の開始
★.pause()	再生の停止
★.load()	メディア(音声・動画ファイル)の読み込み
★.canPlayType(◆)	再生可否を判定する

★……video要素またはaudio要素への参照
◆……MIMEタイプ

形式 メソッド

video要素とaudio要素でのメディアの再生を制御します。APIは2つの要素で共通のものを使用します。

playメソッドでメディアを再生し、pauseメソッドで再生を停止します。再生するメディアを動的に変更した際には、再生の前にloadメソッドで、明示的に読み込みを行う必要があります。

特定のMIMEタイプのメディアが再生可能か判断するには、canPlayTypeメソッドを利用します。再生できないときは空文字が、再生できる可能性が高いときは「probably」が、再生できる可能性はあるが不明なときは「maybe」が返されます。

判定が不確実なのは再生可否にはMIMEタイプ以外の条件も関係するからです。以下のように、ファイルのコーデック(エンコード方式)を含めて引数に指定することもできます。

「video/mp4; codecs="avc1.42E01E, mp4a.40.2"」(mp4形式、動画H.264(AVC1)、音声AAC LC(MP4A)の場合)

メディア要素では以下のようなイベントが発生します。ただしonXXX形式のイベントハンドラは定義されていないため、addEventListenerメソッドを使用する必要があります。サンプルを参照ください。

イベント	内容
emptied	メディアがクリアされた時
loadedmetadata	メタデータが読み込まれた時
loadeddata	データが読み込まれた時
canplay	再生の準備ができた時
canplaythrough	最後まで再生可能になった時
playing	再生待ち状態が終わった時
waiting	次フレームの再生待ち状態の時
ended	再生が完了した時

イベント	内容
durationchange	全体の再生時間が変わった時
timeupdate	再生位置が変わった時
play	停止状態から再生状態になった時
pause	再生状態から停止状態になった時
ratechange	再生レートが変わった時
volumechange	ボリュームが変わった時

※メタデータには再生時間の長さなどの情報が含まれます。

文例

```
if(videoElement.canPlayType("video/webm") == ""){
  output.innerHTML = "webm形式の動画は再生できません";
}
```

webm形式の動画を再生できるか判定しています。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	×	○	○	○	○	○	○

参照
音声・動画の状態を取得したい ………… P.360
【SAMPLE】動画を操作する ………… P.366

音声・動画の状態を取得したい

★.currentSrc	現在のメディアのURL。読み取り専用
★.currentTime	現在の再生位置
★.duration	再生時間の長さ。読み取り専用
★.ended	再生終了判定。読み取り専用
★.muted	ミュート判定
★.paused	停止判定。読み取り専用
★.seeking	シーク中判定。読み取り専用
★.volume	ボリューム(0.0〜1.0)

★……video要素またはaudio要素への参照

形式 プロパティ

video要素またはaudio要素でのメディアの再生時の状態を取得します。

currentSrcプロパティ

現在のロードされているメディアのURLが格納されています。読み取り専用です。

currentTimeプロパティ

現在の再生位置の秒数を取得・設定します。

durationプロパティ

現在のメディアの全体の再生時間が秒数で格納されています。読み取り専用です。

endedプロパティ

再生が終了していればtrueを返します。読み取り専用です。

mutedプロパティ

ミュート(音声無効化)中であればtrueを返します。設定も可能です。

pausedプロパティ

再生が停止中であればtrueを返します。読み取り専用です。

seekingプロパティ

再生位置を変更中であればtrueを返します。読み取り専用です。

volumeプロパティ

ボリュームを0.0～1.0の範囲で取得・設定します。

ここで挙げたもの以外にも、autoplay、controls、loop、preload、srcの各プロパティが存在し、videoまたはaudioタグの同名の属性の値を取得・設定することが可能です。

文例

```
videoElement.muted = true;
```

video要素の再生で、音声を無効にしています。

```
audioElement.currentTime = audioElement.duration / 2;
```

audio要素の再生位置を全体の半分の位置に変更しています。

▶ ブラウザ対応表	IE10	IE9	[illegible]	Fx	Chrome	Safari	[illegible]	[illegible]	Android
	○	○	×	○	○	○	○	○	○

参照

音声・動画の再生を操作したい ………… P.358
【SAMPLE】動画を操作する ………… P.366

GRAPHICS&MEDIA.SAMPLE-01

Canvasに描画する

Canvasへの描画を行っています。コンテキストを取得したのち、フォントを指定し、実際の描画処理は関数に分けています。Canvasの枠線はCSSで指定しています。

drawDirect関数

JavaScript

```
function drawDirect(c) { // パスを使わず直接描画を行います ──① 
    c.save();
    c.fillStyle = "green"; // 塗りつぶし色の設定
    c.fillRect(100, 100, 100, 100); // 塗りつぶしの四角形の描画
    c.strokeStyle = "blue"; // 線の色を設定
    c.globalAlpha = 0.3; // 透明度の変更
    c.strokeRect(50, 50, 100, 200); // 線で四角形を描画
    c.globalAlpha = 1; // 透明度の変更
    c.fillStyle = "black";
    c.fillText("drawDirect", 70, 90); // 文字列の描画
    c.restore();
}

function drawByPath(c) { // パスを使って描画を行います ──②
    c.save();
    c.beginPath(); // パスの開始
    c.moveTo(250, 200); // [illegible]
    c.lineTo(200, 300); // 線を[illegible]
    c.lineTo(300, 450); // 線を[illegible]
    c.closePath(); // 線を閉じる
    c.strokeStyle = "red"; // 線の色を変更
    c.stroke(); // [illegible]で描画
    c.fillText("drawByPath", 250, 250); // 文字列の描画
    c.restore();
}

function drawGradation(c) { // グラデーション[illegible] ──③
    c.save();
    c.beginPath(); // パスのリセット
    c.arc(300, 100, 80, 0, Math.PI * 2); // パスに円の追加
    // 線形グラデーション作成
    var grd = c.createLinearGradient(200, 100, 400, 100);
    grd.addColorStop(0, "white"); // 起点の色を白に設定
```

```
    grd.addColorStop(1, "black"); // 終点の色を黒に設定
    c.fillStyle = grd; // グラデーションを塗りつぶしに設定
    c.fill(); // 塗りつぶしでパスを描画
    c.fillStyle = "red";
    c.fillText("drawGradation", 300, 50); // 文字列の描画
    c.restore();
}

function drawTrans(c) { // 変形を適用します ―――― 4
    c.save();
    c.translate(100, 300); // 移動の変形を追加
    c.scale(1.5, 1.5); // 1.5倍の拡大変形を追加
    c.rotate(45 * Math.PI / 180); // 45度の回転変形を追加
    c.fillStyle = "blue";
    c.globalAlpha = 0.3;
    c.fillRect(10, 10, 70, 70);// 正方形を描画
    c.fillStyle = "green";
    c.fillText("drawTrans", 10, 5); // 文字列の描画
    c.restore();
}

function drawPicture(c) { // 画像を描画します ―――― 5
    var pict = new Image();
    pict.src = "sample_blue.jpg";
    pict.onload = function() { // 画像の読み込みを待つ
        c.save();
        c.drawImage(pict, // 画像ファイルを指定
            250, 250, 100, 100, // 切り抜く領域を指定
            350, 350, 100, 100); // 描画先の領域を指定
        c.fillText("drawPicture", 350, 340); // 文字列の描画
        c.restore();
    };
}

function drawShadow(c) { // 影を付けます ―――― 6
    c.save();
    c.fillStyle = "blue";
    c.shadowColor = "silver"; // 影の色指定
    c.shadowOffsetX = 10; // 影の横の位置
    c.shadowOffsetY = -10; // 影の縦の位置
    c.shadowBlur = 5; // 影の滲み
    c.fillRect(400, 200, 50, 50);
    c.fillStyle = "black";
    c.fillText("drawShadow", 350, 190); // 文字列の描画
    c.restore();
}
```

HTML

※レイアウトはCSSで指定しています

```
<body>
    <canvas id="myCanvas"></canvas>
</body>
```

①drawDirect関数ではパスを使わない図形の描画を行っています。関数の最初でsaveメソッドを呼び、最後でrestoreメソッドを呼んでいるのは、関数内で行った状態変更をクリアするためです。透明度の変更と文字列の描画も行っています。

②drawByPath関数ではパスを使った描画を行っています。

③drawGradation関数では、パスを利用して円を描き、その塗りつぶしとして線形グラデーションを指定しています。

④drawTrans関数では、移動、拡大、回転の変形を加えて、四角形を描画しています。文字列も一緒に変形されています。

⑤drawPicture関数では、画像ファイルを読みこんで、その一部を切り抜いたものを描画しています。新たに画像オブジェクトを作る場合は特に、画像の読み込みを待ってから描画する必要があることに注意してください。

⑥drawShadow関数では影を指定して四角形を描画しています。文字列にも影が付いています。

どの関数で描画したものかを文字列で表示しています

参照 関連メソッド・プロパティ ……… P.350～356

GRAPHICS&MEDIA.SAMPLE-02

動画を操作する

video要素で埋め込んだ動画の再生、停止ボタンと、現在の再生時点を示すスライダーを、既定のコントロールを使わずに実装するサンプルです。現在の再生時点の変更は、再生するにつれて発生するtimeupdateイベントで検知し、currentTimeプロパティを使って取得と設定を行います。スライダーの値を変更すると再生位置も変わります。

JavaScript

```
var v = null;
window.onload = function() {
    // 必要なHTML要素への参照を取得
    v = document.getElementById("v");
    var slider = document.getElementById("slider");
    var disp = document.getElementById("disp");

    // スライダーの最大値を再生時間に設定
    slider.max = v.duration;
    v.addEventListener("durationchange", function() {
        slider.max = v.duration;
    });

    // 再生時点からスライダーへの同期を設定
    v.addEventListener("timeupdate", function() {
        slider.value = v.currentTime;
        disp.innerHTML = Math.ceil(v.currentTime) + " / " + Math.ceil(v.duration);
    });

    // スライダーから再生時点への同期を設定
    slider.addEventListener("change", function() {
        v.currentTime = slider.value;
    });
};
```

HTML

※レイアウトはCSSで指定しています

```
<body>
<form>
  <video id="v">
    <source src="sample.webm" type="video/webm">
    <source src="sample.mp4" type="video/mp4">
  </video>
  <div>
    <input id="slider" type="range" value="0">
    <div id="ctlr">
      <span id="disp"></span>
      <input onclick="v.play();" type="button" value="再生" >
      <input onclick="v.pause();" type="button" value="停止">
    </div>
  </div>
</form>
</body>
```

Internet Explorer

ボタンで動画の再生・停止。シークバーで再生位置の指定ができます

Firefox

メディア操作のサンプル

12.08

13 / 13 再生 停止

Opera

Android

iphone

参照
duration プロパティ ·················· P.360
currentTime プロパティ ················ P.360
play メソッド ························ P.358
pause メソッド ······················· P.358

FILE HANDLING.01

ドラッグ&ドロップできるようにしたい

```
★.ondragstart = ◆                      ドラッグ開始時
▲.ondragenter = ◆                      ドロップ先要素への[illegible]触[illegible]
▲.ondragover = ◆                       ドロップ先[illegible]素上での[illegible]時
▲.ondrop = ◆                           ドロップ時
●.dataTransfer.setData(■, ▼)           データの格納
●.dataTransfer.getData(■)              データの取出し
```

★……ドラッグ&ドロップで移動する[illegible]
◆……実行する命令(関数や[illegible]名)
▲……ドラッグ&ドロップの移動先の[illegible]
●……イベント引数
■……データのタイプ
▼……データ

形[illegible] イベント(ondragstart、ondragenter、ondragover、ondrop)
メソッド(setData、getData)

ドラッグ&ドロップでは、或る要素を別の要素に引き渡すことで、コピー、移動、リンク、その他の処理を行います。

要素をドラッグ(マウスカーソル[illegible]で動かすこと)可能にするには、HTMLソースでdraggable属性をtrueに設定します。[illegible]や[illegible]文字列などは既定でドラッグ可能です。

要素をドロップ(他の要素を受け付けること)可能にするには、仕様ではドロップ先の要素にdropzone属性(値はドロップ時の処理に応じて「copy」「move」「link」のいずれか)を設定します。ただし、2013年3月現在、dropzone属性を実装しているブラウザがないため、以下の方法で代替します。

ブラウザによるドロップの禁止は、dragenterイベント、dragoverイベントの既定の処理で行われています。その為、この「既定のイベント処理」をキャンセルすることで、ドロップを有効にすることができます(コラム p.372 及びサンプル p.376 参照)。

ドラッグされた要素からドロップ先の要素へデータを渡すには、DataTransferオブジェクトを仲立ちにします。

データを格納するには、dragstartイベントで[イベント引数].dataTransfer.setData(■, ▼)メソッドを呼び出します。1つのタイプ■につき、1つのデータ▼が格納可能です。既定でドラッグ可能な要素では、あらかじめ適切なデータが対応するタイプに格納されます。

データのタイプ指定文字列■は、「text/plain」(文字列)「text/uri-list」(URL)などのMIMEタイプで指定しますが、古い仕様の「Text」「URL」も使用可能で、IEは「Text」「URL」にのみ対応しています。

データを取り出すには、[イベント引数].dataTransfer.getData(■)を適切なタイプ■を引数にして呼び出します。

文例

```
dropElement.ondragenter = function(event){
    event.preventDefault();
}
```

dragenterイベントについて、ドロップ禁止を解除します。

```
dropElement.ondrop = function(event){
    var data = event.dataTransfer.getData("Text");
}
```

ドラッグ&ドロップによって渡された文字列を取得します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	×	×	○	○	○	○	×	×

ブラウザ外とのドラッグ&ドロップのやり取りをしたい・・P.372
[SAMPLE] ドラッグ&ドロップで要素の色を変える・・P.376

ブラウザ外とドラッグ&ドロップのやり取りをしたい

◆ = ★.dataTransfer.files[参照番号]　ドラッグ&ドロップされたファイルの取得

★……ドロップ時イベントのイベント引数
◆……ファイルの情報を表すFileオブジェクト

プロパティ

要素がドロップ可能であれば、ブラウザ外からドラッグされたファイルや要素を受け付けることも可能です。この際、ファイルへのページ遷移という既定の動作を回避するには、dropイベントで「既定のイベント処理」をキャンセルします(コラム参照)。

ファイルの場合、DataTransferオブジェクトのfilesプロパティに、Fileオブジェクトが配列形式で格納されます。Fileオブジェクトはファイルの中身ではなく、属性情報を格納しています。ファイルの中身の取得にはFile API(p.373参照)を使用します。

文例

```
var file = event.dataTransfer.files[0];
```
ドラッグ&ドロップされたファイルを表すFileオブジェクトを取得します。

```
var name = file.name;
```
ファイル名を取得します。

Column　既定のイベント処理のキャンセル

ブラウザは各イベントに対応してさまざまな処理を既定で行っています。そのうち、キャンセル可能な処理については、各イベントで[イベント処理関数の引数].preventDefault()を呼び出すことで無効にすることができます。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	×	×	○	○	○	○	×	×

参照
ドラッグ&ドロップできるようにしたい……P.370
ファイルの属性を取得したい……P.373
ファイルの内容を取得したい……P.374
【SAMPLE】ドラッグ&ドロップで要素の色を変える……P.376

FILE HANDLING.03

ファイルの属性を取得したい

★ = ◆.files[参照番号]	Fileオブジェクトの取得
★.name	ファイル名
★.size	サイズ(単位はバイト)
★.type	ファイルのMIMEタイプ
★.lastModifiedDate	最終更新日時(Dateオブジェクト)

★……Fileオブジェクト
◆……input要素またはDataTransferオブジェクト

書式 プロパティ

HTML5ではJavaScriptからファイルにアクセスすることが可能になりました。ファイルを表すFileオブジェクトの取得には、type属性がfileのinput要素を使う方法と、ドラッグ&ドロップを使う方法があります。

input要素のファイルダイアログで選択すると、changeイベントが発生し、input要素のfiles属性にFileオブジェクトが配列形式で格納されます。取得の際には、files[0]のように、参照番号を指定します。

ドラッグ&ドロップの場合は、dropイベントで、「[イベント引数].dataTransfer.files」に、同じようにFileオブジェクトが格納されます(p.372参照)。

Fileオブジェクトは、ファイル情報の取得のほかに、ファイルの中身を読み込む際の引数としても使用します。

文例

```
var file = inputElement.files[0];
```
input要素からFileオブジェクトを取得します。
```
var type = file.type;
```
ファイルのMIMEタイプを取得します。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	×	×	○	○	○	○	○	○

ブラウザ外とのドラッグ&ドロップのやり取りをしたい……P.372
ファイルの内容を取得したい……P.374
【SAMPLE】ファイルの内容を取得する……P.379

FILE HANDLING.04

ファイルの内容を取得したい

★.readAsArrayBuffer(◆)	ArrayBufferオブジェクトとして取得
★.readAsText(◆, ▲)	文字列として取得
★.readAsDataURL(◆)	Data URLとして取得
★.abort()	読み込みを中止
★.result	ファイルから読み込まれたデータ

★……FileReaderオブジェクト
◆……Fileオブジェクト
▲……文字列のエンコーディング

形式 メソッド

ファイルの中身を読み込むには、FileReaderオブジェクト(「new FileReader()」で生成)を使用します。なお、ArrayBufferとData URLの使用法については、サンプルを参照してください。

readAsArrayBufferメソッド

ファイルの内容を、型付きデータ配列のためのArrayBufferオブジェクトとして格納します。

readAsTextメソッド

ファイルの内容を、指定された文字コード(IANA名。省略可)のテキストデータとして文字列に格納します。

readAsDataURLメソッド

ファイルの内容を、Data URLに変換して文字列に格納します。

各メソッドは読み込み処理の起動だけを行い、読み込みの完了を待たずに終了します(非同期)。バックグラウンドで読み込みが完了すると、FileReaderオブジェクトのresult属性にデータが格納され、loadイベントが発生します。

FileReaderオブジェクトの読み込みに関するイベントには下記のようなものがあります。

イベント	説明	イベント	説明
loadstart	読み込み開始	load	読み込みが正常終了
progress	読み込み中	error	読み込みが異常終了
abort	読み込み中止	loadend	読み込みが成功失敗にかかわらず終了した

文例

```
fReader.addEventListener("load", function(ev){ output.innerHTML = fReader.
result; });
```

ファイル読み込み完了時の処理を定義しています。

```
fReader.readAsText(file, "utf-8");
```

テキストファイルの読み込みを非同期で開始しています。

ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	×	×	○	○	○	○	○	○

ブラウザ外とのドラッグ＆ドロップのやり取りをしたい・・P.372
ファイルの属性を取得したい・・・・・・・・・・・・・P.373
【SAMPLE】ファイルの内容を取得する・・・・・・P.379

FILE HANDLING.SAMPLE-01

ドラッグ&ドロップで要素の色を変える

左側のボックスのどちらかを右端のボックスにドラッグ&ドロップすると、動かしたボックスと同じ文字色・背景色に変化します。draggable属性でドラッグを許可し、各イベントを定義してドロップを許可し、要素のidを渡すことで、元の要素のスタイルを取得しています。getComputedStyle関数は、最終的に要素に適用されたスタイルを取得します。なお、draggable属性はIE9までのIEは対応していません。

HTML&JavaScript

※レイアウトはCSSで指定しています

```
<body>
<div class="box" id="drag_01" draggable="true">ドラッグされる要素01</div>
<div class="box" id="drag_02" draggable="true">ドラッグされる要素02</div>
<div class="box" id="drop_el">ドロップ先要素</div>

<script>
  // 要素を取得します
  var drag_01 = document.getElementById("drag_01");
  var drag_02 = document.getElementById("drag_02");
  var drop_el = document.getElementById("drop_el");
  // ドラッグ開始時のイベント処理関数を定義します
  function dragHandler(e) {
    // ドラッグされた要素のidを格納します
    e.dataTransfer.setData("Text", e.target.id);
  }
  // ドラッグ開始時のイベント処理関数を登録します
  drag_01.addEventListener("dragstart", dragHandler);
  drag_02.addEventListener("dragstart", dragHandler);

  // 既定のイベント処理を無効にしてドロップを有効にします
  drop_el.addEventListener("dragenter", function(e) {
    e.preventDefault();
  });
  drop_el.addEventListener("dragover", function(e) {
    e.preventDefault();
  });

  // ドロップ時のイベント処理を定義します
  drop_el.ondrop = function(e) {
  // ドラッグされた要素のidを取り出します
```

```
        var id = e.dataTransfer.getData("Text");
        // idから要素を取得します。
        var el = document.getElementById(id);
        // 要素のCSS適用後の実際のスタイルを取得します。
        var style = getComputedStyle(el);
        // 背景色と文字色をドラッグされた要素のものに設定します。
        drop_el.style.backgroundColor = style.backgroundColor;
        drop_el.style.color = style.color;
    };

</script>
</body>
```

CSS

```
.box{
    height: 100px;
    width:100px;
    border:solid 1px black;
    margin:20px 20px 50px 20px;
    padding:5px;
    float:left;
}
#drag_01{
    background-color:green;
    color:orange;
}
#drag_02{
    background-color:blue;
    color:red;
}
```

Internet Explorer

ドラッグ＆ドロップされた要素と同じ文字色・背景色になります。

参照
dataTransfer.setData メソッド ･･････････ P.370
dataTransfer.getData メソッド ･･････････ P.370
onDrop イベント ･･････････････････････ P.370

FILE HANDLING.SAMPLE-02

ファイルの内容を取得する

ファイルダイアログで取得したファイルのファイル名とMIMEタイプを表示し、MIMEタイプで読み込み後の処理を分けています。この際、直感的な順序とは逆ですが、非同期処理では、完了時の処理をまず定義し、その後で読み込みを開始する必要があります。

画像ファイルなら、ファイルの中身を読み込んでData URL形式に変換し、画面に表示しています。Data URL形式はURLに実際のデータを埋め込むための形式で、img要素のsrc属性に設定すると、画面にその画像データを表示することができます。

テキストファイルならテキストエリアを作成して、その中に内容を表示しています。

それ以外のファイルタイプの場合は、ArrayBuffer形式で読み込んだうえで、最初の1バイト目の値を表示しています。型付き配列用のArrayBufferオブジェクトの使用法はサンプルの該当箇所を参照してください。

HTML

※レイアウトはCSSで指定しています

```
<body>
<form>
  <input id="fileInput" type="file" >
  <div id ="message"></div>
  <div id="output"></div>
</form>
</body>
```

JavaScript

```
  要素の参照を取得します
var fileInput = document.getElementById("fileInput");
var output = document.getElementById("output");
var message = document.getElementById("message");

  ファイルが選択された際の処理を定義します
fileInput.onchange = function(event) {

  選択された最初のファイルを取得します
var file = fileInput.files[0];
```

```
if (!file) {
    // キャンセル時などのためにNULLチェックを行います
    return;
}

// ファイルの情報を出力します
output.innerHTML = "";
output.innerHTML += "ファイル名:" + file.name;
output.innerHTML += "<br>MIMEタイプ:" + file.type;
output.innerHTML += "<hr>";

// FileReaderオブジェクトを生成します
var fileReader = new FileReader();

// 読み込み開始時の処理を定義します
fileReader.onloadstart = function() {
    message.innerHTML = "読み込み中です。";
};

// 読み込み全体の終了時の処理を定義します
fileReader.onloadend = function() {
    message.innerHTML = "";
};

// MIMEタイプで読み込み処理を分岐します
if (/image¥/.*/.test(file.type)) {

    /* 画像ファイルの場合 */

    // 読み込み完了時の処理を登録します
    fileReader.onload = function(event) {
        var image = new Image();
        // Data URL形式の画像データをimg要素に設定します
        image.src = fileReader.result;
        output.appendChild(image);
    };

    // 画像ファイルをData URL形式で読み込みます
    fileReader.readAsDataURL(file);
} else if (/text¥/.*/.test(file.type)) {

    /* テキストファイルの場合 */

    // 読み込み完了時の処理を登録します
    fileReader.onload = function(event) {
        var textarea = document.createElement("textarea");
```

```
            // テキストエリアに読み込んだテキストを設定します
            textarea.value = fileReader.result;
            output.appendChild(textarea);
        };

        // shift_jisとして読み込みます
        // テキストが指定の文字コードで無い場合は文字化けします
        fileReader.readAsText(file, "shift_jis");

    } else {

        /* それ以外の場合 */

        fileReader.onload = function(event) {
            var div = document.createElement("div");
            // ArrayBuffer形式のデータを取得します
            var buffer = fileReader.result;
            // バイト配列(Uint8Array)に変換します
            var byteArray = new Uint8Array(buffer);
            // 最初の1バイトの値を取得します
            div.innerHTML = "このファイルの最初の1バイトの値は " +
                                    byteArray[0] + " です";
            output.appendChild(div);
        };

        // ArrayBuffer形式で読み込みます
        fileReader.readAsArrayBuffer(file);
    }

};
```

ファイルの内容を取得する

【ファイル操作】

Internet Explorer

画像ファイルの場合、読み込んだ画像を表示します。

テキストファイルの場合、テキストエリア内にテキストを表示します。

それ以外のファイルの場合、読み込んだ最初の1バイト目の値を表示します。

Android

iphone

参照

files プロパティ・・・・・・・・・・・・・・・・・・・・ P.373
name プロパティ・・・・・・・・・・・・・・・・・・・・ P.373
type プロパティ・・・・・・・・・・・・・・・・・・・・ P.373
readAsDataURL メソッド・・・・・・・・・・・・・・ P.374
readAsText メソッド・・・・・・・・・・・・・・・・ P.374
readasArrayBuffer メソッド・・・・・・・・・・・・ P.374

LOCALDATA&OFFLINE.01

ブラウザの保存領域にデータの読み書きを行いたい

★ = window.sessionStorage	セッション・ストレージの取得
★ = window.localStorage	ローカル・ストレージの取得
★.length	格納されている「キー／値」ペアの[illegible]の取得
★.key(参照番号)	参照番号の位置にあるキーの[illegible]
★.getItem(◆)	キー◆に対応する値の取得
★.setItem(◆,▲)	キー◆に対応する値▲の設定
★.removeItem(◆)	キー◆とその値の削除
★.clear()	すべてのキーと値の削除

★……Storageオブジェクト
◆……データが格納される項目名(キー)
▲……項目に保存する値(文字列)

形式 プロパティ(sessionStorage、localStorage、length)
メソッド(key、getItem、setItem、removeItem、clear)

HTML5ではデータをユーザー環境に保存する手段として、ローカル・ストレージとセッション・ストレージが利用可能です。

ローカル・ストレージはオリジン(コラム参照)ごとに独立したデータの保存領域を用意するもので、オリジンが同一ならウィンドウやページをまたいで共有されます。また、明示的に削除しない限り、ブラウザを終了させても内容が保持されます。

セッション・ストレージはセッションごとにデータの保存領域を用意するもので、セッションの終わりとともに破棄されます。また、セッションが同一でもオリジンが異なるとセッション・ストレージは別になります。

一つのセッションは、そのウィンドウ(タブに分かれている場合はそれぞれのタブ)が開いているあいだ存続します。同じウィンドウまたはタブのなかでページ遷移しても、セッションは変わりません。

フレームは自分が属しているページとセッションを共有します。子ウィンドウは、親のセッションの内容をコピーして、新しいセッションを開始します。

データを読み書きする方法はストレージで共通です。データに項目名(キー)を割り当てて保存し(setItemメソッド)、取得するときにはそのキーで取り出します(getItemメソッド)。

key(参照番号)メソッドは「参照番号」の位置に格納されたキーを返しますが、キーの位置を決めるルールはブラウザの実装に任されています。値の更新では位置は変わりませんが、追加や削除の場合は変わることがあります。通常は、for文ですべてのキーを列挙するような場合に使用します。

ローカルファイルを「file://」で始まるURLやファイルパスで開いた場合は、ブラウザの実装によって、ストレージが存在しない、storageイベントが発生しないといった問題があるため、テストの際には注意が必要です。

文例

```
window.sessioStorage.setItem("age", 23);
```
セッション・ストレージにキー「age」で値「23」を保存します。

```
var userAge = window.sessionStorage.getItem("age");
```
セッション・ストレージからキー「age」の値を取得します。

Column オリジン(origin)

オリジンは送信元とも訳され、URLの「スキーム」「ドメイン」「ポート」を合わせたものです。たとえば、

http://www.example.com:8080/dir/index.html

というURLでは、「http://www.example.com:8080」までがオリジンです。

オリジンはスクリプトによるアクセスを制限する際の単位として使用されます。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	[illegible]	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

データ変更イベントをハンドリングしたい …… P.386
【SAMPLE】テキストの内容をWeb Storageに保存する …… P.394

LOCALDATA&OFFLINE.02

データ変更イベントをハンドリングしたい

window.onstorage = ★

★……実行する命令（関数や[illegible]名）

形式 イベント

Web Storageでは、[illegible]のウィンドウ・タブが同一のデータ保存領域（ストレージ）に対して読み書きを行うことが可能です。

そのため、別のウィンドウ・タブで行われた変更を検知して適切に対処しなければ、データの整合性に問題が生じる場合があります。

データの変更の検知には、onstorageイベントを利用します。このイベントは、変更が行われたストレージを共有するウィンドウやタブの、windowオブジェクトで発生しますが、IE以外では、変更を行ったウィンドウやタブそのものでは発生しません。

onstorageイベントでは、イベントの引数の属性として次のような情報が取得できます。

プロパティ	説明
key	変更されたキー
oldValue	変更前の値
newValue	変更後の値
url	変更したページのURL
storageArea	変更されたストレージ・オブジェクト

文例

```
window.onstorage = function(event){
  var key = event.key;
  console.log(key);
}
```

値が変更されたキーをデバッグ用コンソールに出力しています。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	[illegible]	[illegible]	Android
	○	○	○	○	○	○	○	○	○

参照
ブラウザの保存領域にデータの読み書きを行いたい‥ P.384
【SAMPLE】テキストの内容を Web Storage に保存する ‥‥‥ P.394

Indexed DBへの接続や初期化をしたい

★ = window.indexedDB.open(◆, ▲)　データベースへの接続
★ = window.indexedDB.deleteDatabase(◆)　データベースの削除
■.close()　データベースへの接続終了

★……リクエスト・オブジェクト。
◆……データベース名
▲……接続するデータベースのバージョン(1以上の整数値)
●……実行する命令(関数や関数名)
■……データベース・オブジェクト

形式 メソッド

HTML5では簡単なキー/値ペアでのデータの保存にはWeb Storage(p.384参照)が用意されていますが、大量のデータや高度な機能を扱うには、Indexed DBを使用する必要があります。

Indexed DBでは、データベースの中に任意の数だけ作成可能な、「オブジェクトストア」(RDBMSのテーブルに相当)に。任意の数のデータをオブジェクトとして格納します。

また、Indexed DBでは、非同期処理が採用されています。メソッドは、バックグラウンド処理の起動だけを行い、リクエスト・オブジェクト★を返して終了します。その後、処理が完了すると、successイベントが戻り値★で発生し、結果が★.resultに格納されます。

★ = window.indexedDB.open(◆, ▲)
★.onsuccess = ●

データベースに接続するには、データベース名◆とバージョン▲を指定します。接続成功時には非同期リクエスト★のresult属性にデータベース・オブジェクトが格納されます。

指定した名前のデータベースが存在しなければ新規に作成され、指定したバージョンより接続先のバージョンが上だった場合には接続が失敗します。この時、作成・接続できるデータベースは、ページとオリジン(p.385参照)が同じものに限定されます。

★.onupgradeneeded = ■

接続先のバージョン(新規作成時は0)より上のバージョンを指定した場合、successイベントの前にupgradeneededイベントが発生します。このイベントは、初期化とアップグレード処理のためのイベントです。オブジェクトストアとインデックスの作成・削除はこのイベントの中でしか行えません。

▼ = ■.createObjectStore(☆, ◇);

データベースにオブジェクトストアを作成するには、名前☆と、keyPath属性とautoIncrement属性を持つオブジェクト◇を指定します。

keyPath属性には、オブジェクトストアが、オブジェクトを取得する際にキーとして使用する属性名を指定します。autoIncrement属性には、オブジェクトを格納する際、キー属性に番号をカウントアップしつつ設定するかどうかを指定します。

upgradeneededイベントの中でデータの追加や加工を行うこともできます。ただし、このイベントの中では、★.transactionから取得できる、"versionchange"モードのトランザクション(次項で解説)がすでに開始されているため、新たにそれ以外のトランザクションを開始することはできません。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	×	×	○	○	×	×	×	×

参照
データの追加、更新・削除がしたい ········ P.389
【SAMPLE】Indexed DB を操作する ······· P.397

データの追加、更新・削除がしたい

★ = ◆.transaction(▲, ●)　　トランザクション[illegible]
■ = ★.objectStore(▼)　　オブジェクトストア取得

★……トランザクション・オブジェクト
◆……データベース・オブジェクト
▲……対象になるオブジェクトストア名の配列
●……トランザクションのモード
■……オブジェクトストア・オブジェクト
▼……オブジェクトストア名

形式　メソッド

Indexed DBではデータの読み書きはトランザクションを通じて行います。トランザクションで行われたデータの変更は、トランザクション単位で適用されます。そのため、変更が半端に適用され、データの整合性が壊れるのを防ぐことができます。

★ = ◆.transaction(▲, ●)

データの読み書きを行うには、まず、データベース・オブジェクト◆のtransactionメソッドでトランザクションを開始します(upgradeneededイベントを除く)。

配列▲は、トランザクションでアクセスするオブジェクトストア名の配列です。モード●は開始するトランザクションが読み書きモードなら"readwrite"を、読み取り専用モードなら、"readonly"を指定します。読み書きモードのトランザクションは、同時に一つしか存在できません。

■ = ★.objectStore(▼)

実際にデータにアクセスするには、トランザクション★のobjectStoreメソッドでオブジェクトストアを取得します。引数▼は対象のオブジェクトストアの名前です。

☆ = ■.add(◇[, △]) ／ ☆ = ■.put(◇[, △])

オブジェクトストア■にデータ◇を追加するにはaddメソッドかputメソッドを使用します。オブジェクトストアにキー属性が設定されていない場合はキー△も指定します。キーが重複していた場合、addではエラーになりますが、putでは上書きになります。

オブジェクト◇はそのものではなく、コピーが格納されます。DOM要素などの、特定のウィンドウやタブを越えてコピー不可能なタイプのオブジェクトは保存できません。

☆ = ■.get(△) ／ ☆ = ■.delete(△) ／ ☆ = ■.clear()

取得にはgetメソッドを、削除にはdeleteメソッドをキー△を指定して呼び出します。clearメソッドですべてのデータが削除されます。

処理成功時には、リクエスト・オブジェクト☆のresult属性に、add/putメソッドでは追加・更新されたキーが、getメソッドでは取得したオブジェクトが格納されます。

○ = ■.index(□)

インデックス○を取得するにはインデックス名□を指定してindexメソッドを呼び出します。upgradeneededイベント内で、予め作成しておく必要があります。

インデックスには、オブジェクトストアと同じget(△)メソッドが存在し、オブジェクトストアのキー属性(プライマリー・キー)とは別に、インデックス独自のキー属性をキーにしてオブジェクトを取得できます。

また、getメソッド、deleteメソッドで指定するキーは通常のキーのほかに「キー範囲」も指定できます。詳細はサンプルをご覧ください。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	[illegible]	Android
	○	×	×	○	○	×	×	×	×

参照

Indexed DB への接続や初期化をしたい・・・・・P.387
【SAMPLE】Indexed DB を操作する・・・・・・・P.397

LOCALDATA&OFFLINE.05

オフライン時にもキャッシュを表示させたい

<html manifest=★>　マニフェスト・ファイルの指定

★……キャッシュ・マニフェストのパス

形式 HTML属性値

アプリケーション・キャッシュでは、自動的に保存された控え（キャッシュ）を、オフライン時に代替として使用するように設定することができます。

キャッシュを利用するには、マニフェスト・ファイルを作成し、html要素のmanifest属性にそのパスを指定します。マニフェストの最初の行には「CACHE MANIFEST」とだけ記載します。

マニフェストは「CACHE:」「NETWORK:」「FALLBACK:」のいずれかで始まるセクションに分かれ、それぞれのセクションで、「キャッシュ対象の指定」「キャッシュ除外の指定」「代替コンテンツの指定」を行います。「#」で始まる行はコメント行です。詳細はサンプルを参照ください。

Sample (cache.manifest)

```
CACHE MANIFEST

# セクション無指定の場合、既定では「CACHE:」セクションになります。
sample.html

NETWORK:
network.html
```

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

現在のキャッシュ状態を取得したい ……… P.392
【SAMPLE】オフライン状態とキャッシュ状態を取得する ……… P.401

現在のキャッシュ状態を取得したい

★ = window.applicationCache	キャッシュ・オブジェクトの取得
★.update()	キャッシュの更新(ダウンロードのみ)
★.swapCache()	update()でダウンロードしたファイルの適用

★……アプリケーション・キャッシュ・オブジェクト

形式 プロパティ(applicationCache)、メソッド(update、swapCache)

読み込み時にマニフェストが更新されていると、キャッシュも更新されます。その際の更新状況は、window.applicationCacheのstatus属性から取得できます。

定数名	値	状態	定数名	値	状態
UNCACHED	0	キャッシュしていない	DOWNLOADING	3	キャッシュのダウンロード中
IDLE	1	キャッシュは最新	UPDATEREADY	4	ダウンロード完了（未適用）
CHECKING	2	マニフェストのチェック中	OBSOLETE	5	マニフェストが存在しなくなった

updateメソッドで明示的に更新を開始させることもできます。ただし、swapCacheメソッドで適用するまで、新規にダウンロードしたファイルはキャッシュに反映されません。

各キャッシュ状態は、window.applicationCacheに以下のイベントを発生させます。

イベント	説明	イベント	説明
checking	マニフェストのチェック中	downloading	ダウンロード開始
noupdate	マニフェストの更新なし	progress	ダウンロード中
obsolete	マニフェストが存在しなくなった	updateready	ダウンロード完了（未適用）
error	マニフェストがない等のエラー	cached	キャッシュ更新完了

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	×	×	○	○	○	○	○	○

オフライン時にもキャッシュを表示させたい‥‥P.391
【SAMPLE】オフライン状態とキャッシュ状態を取得する‥‥P.401

LOCALDATA&OFFLINE.07

オンライン・オフライン状態を取得したい

★ = **window.navigator.onLine** オンライン状態の取得

★……オンライン状態(オンラインならtrue、オフラインならfalse)

形式 プロパティ

ブラウザがネットワークにつながっているかどうかは、ナビゲーター・オブジェクトのonLineプロパティで取得することができます。オンライン時はこの値がtrueになり、オフライン時はfalseになります。

また、オンラインになった時点、オフラインになった時点で、body要素(さらにdocument及びwindowオブジェクト)でonlineイベント、offlineイベントが発生します。

文例

```
document.onoffline = function(){
    alert("オフラインになりました");
}
```

オフライン・イベントを取得しています。

▶ ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS6	Android
	○	×	×	○	○	○	○	○	○

【SAMPLE】オフライン状態とキャッシュ状態を取得する P.401

LOCALDATA&OFFLINE.SAMPLE-01

テキストの内容を Web Storageに保存する

テキストエリアの内容をローカルストレージに保存しています。このサンプルを複数のウィンドウやタブで開き、一方を更新すると、他のすべてのウィンドウ・タブもリアルタイムに同期します。また、いったんブラウザを終了させても、以前のデータが復元されます。

同期については、storageイベントを使って、他のウィンドウ・タブでの変更を検知して再読み込みを行っています。ただし、IEでは、変更を行ったウィンドウ・タブ自身でもstorageイベントが発生するため、その対策も行っています。

変更を行ったウィンドウ・タブ自身で、さらに再読み込みを行うと、タイミングの問題で更新中のデータが失われてしまうことがあります。

そのため、それぞれのウィンドウ・タブが、自分自身のＩＤを生成して、保存するデータにこのＩＤを含めます。読み込み時には、そのデータに記録されているＩＤと自分自身のＩＤを比較して、最後に保存したのが自分だった場合は、読み込みを行わないようにしています。

保存するデータにＩＤを含めるために、cid属性がこのＩＤ，value属性がテキストという保存用のオブジェクトを用意しています。ストレージには基本的に文字列しか保存できないため、保存時にはJSON形式の文字列に変換し(JSON.stringify()メソッド)、読み込み時に復元(JSON.parse()メソッド)しています。

HTML

※レイアウトはCSSで指定しています

```
<body>
<form>
    <textarea id="text"></textarea>
</form>
</body>
```

JavaScript

```
// ストレージに保存する際のキーを定義
var STORAGE_KEY = "webstorage sample";

// このタブのIDを生成(ＩＥ対策)
var CLIENT_ID = (new Date()).getTime().toString();

// テキストエリアへの参照を取得
var text = document.getElementById("text");

// ストレージからの読み込み処理を定義
function load() {
    var data = null;
    // 復元の失敗に備えてtry-catchで囲む
    try {
        // JSON形式で保存したものをオブジェクトに復元
        data = JSON.parse(localStorage.getItem(STORAGE_KEY));

        // タブごとに生成したＩＤを利用して
        // このタブで行った変更は読み込まない(IE対策)
        if (data.cid !== CLIENT_ID) {
            text.value = data.value;
        }

    } catch (e) {

    }
}

// ストレージへの保存処理を定義
function save() {
    // 保存用のオブジェクトを定義
    // cid属性にタブのＩＤを設定(最後に更新したタブを記録)
    var data = {cid: CLIENT_ID, value: text.value};
    // オブジェクトをJSON形式に変換してストレージに保存
    localStorage.setItem(STORAGE_KEY, JSON.stringify(data));
}

// 初回のデータの読み込み
load();

// テキスト入力のたびにストレージを更新するように
// イベントハンドラを設定
text.oninput = save;

// データ変更イベントを定義
```

```
// Firefoxではonstorageイベントハンドラが定義されていないため
// addEventListenerを使用します。
window.addEventListener("storage", function(event) {

    //キーのチェック
    if (event.key !== STORAGE_KEY) {
        return;
    }

    // 他のタブでの変更を読み込んで[illegible]
    load();

});
```

入力したテキストがブラウザに保存されます

同じページを別のタブで開いても変更が反映されます

参照

localStorage プロパティ ………… P.384
getItem メソッド ………… P.384
setItem メソッド ………… P.384

Indexed DBを操作する

「初期化」ボタンで、Indexed DBにデータを格納し、テーブルに表示します。「アップグレード」ボタンを押すと、データベースのバージョンが2になり、upgradeneededイベントでデータが更新され、id列(属性)が追加されます(該当箇所はupgradeHandler関数)。

JavaScript

```
// 初期化・クリア
function clearData() {
    window.indexedDB.deleteDatabase(DB_NAME);
    clearUI(); // ボタンを初期化
    global_data = [ // 初期データ用の配列
        {name: "山田", age: 45, mail: "a@example.com"},
        {name: "田中", age: 15, mail: "b@example.com"},
        {name: "佐藤", age: 20, mail: "c@example.com"}
    ];
}

// データベースへの接続処理
function connectDBAsync(version) {
    // データベースへの接続をリクエスト
    var openReq = window.indexedDB.open(DB_NAME, version);
    // 初期化・アップグレード時
    openReq.onupgradeneeded = upgradeHandler;
    // 接続成功時
    openReq.onsuccess = successHandler;
}

// データベースの初期化・アップグレード処理
function upgradeHandler(event) {
    var tr = event.target.transaction; // トランザクション取得
    var store1;
    switch (event.newVersion) { // バージョンで処理を分岐
        case 1:
            store1 = tr.db.createObjectStore("store1", {keyPath: "name"});
            store1.createIndex("ageIndex", "age");
            for (var i = 0; i < global_data.length; i++) {
                store1.add(global_data[i]); // 初期データ追加(非同期)
            }
            break;
```

```
            case 2:
                // データ更新(非同期)
                store1 = tr.objectStore("store1");
                for (var i = 0; i < global_data.length; i++) {
                    var val = global_data[i];
                    val.id = i;// global_dataの元データを更新
                    store1.put(val);// 更新したデータで上書き
                }
                break;
        }
}

// 接続成功時の処理
function successHandler(event) {
    global_db = event.target.result; // データベースの参照を取得
    global_db.onversionchange = function(e) {
        e.target.close(); // アップグレード時にはいったん切断
    };
    setUI(global_db.version); // ボタンの有効無効切り替え
    displayData(false); // データの表示
}

// データの表示
function displayData(isFilter) {
    var output = document.getElementById("output");
    output.innerHTML = "";
    var tr = global_db.transaction(["store1"], "readonly");
    var store1 = tr.objectStore("store1"); // オブジェクトストア取得
    var cursorReq = null;
    if (isFilter) {
        var ageIndex = store1.index("ageIndex");
        // インデックス取得
        // キー範囲(10から20)を定義
        var keyRange = IDBKeyRange.bound(10, 20);
        cursorReq = ageIndex.openCursor(keyRange); // カーソルをリクエスト
    } else {
        cursorReq = store1.openCursor(); // カーソルをリクエスト
    }
    // 指定した範囲の全件を表示
    cursorReq.onsuccess = function() {
        var cursor = cursorReq.result; // カーソルの取得
        if (cursor) {
            var value = cursor.value; // 現在の位置のデータ取得
            var row = output.insertRow(-1); // tableに行追加
            for (var p in value) { // 取得したデータの全属性を表示
                var cell = row.insertCell(-1); // 行にセル追加
```

```
            cell.innerHTML = p + " : " + value[p]; // 値を設定
        }
        cursor.continue(); // カーソルを次の位置へ進める
    }
  };
}
```

HTML ※レイアウトはCSSで指定しています

```
<body>
<form>
  <input id="init" type="button" onclick="connectDBAsync(1);" value="データベース初期化">
  <input id="upgrade" type="button" onclick="connectDBAsync(2);" value="バージョン2へアップグレード">
  <input id="filter" type="button" onclick="displayData(true);" value="10才から20才にフィルター">
  <input type="button" onclick="clearData();" value="クリア">
  <table id="output"></table>
</form>
</body>
```

「10才から20才にフィルター」ボタンを押すと、age属性に基づいてデータがフィルターされます。この処理では再接続せずに既存の接続をglobal_db変数を介して利用しています。また、表示処理ではインデックス、カーソル、キー[illegible]が使用されています(該当箇所はdisplayData関数)。

age[illegible]をキーにデータを取得するには、ageをキーとしたインデックスを使用します。データを順次列挙するにはカーソルを利用します。[illegible]を限定するにはキー範囲を使用します。「IDBKeyRange.bound(10, 20)」は10から20を意味します。キー[illegible]はgetメソッドやdeleteメソッドの引数にも使用可能です。

カーソルはデータを列挙するためのオブジェクトで、この処理も非同期です。カーソルにはkey属性とvalue属性があり、現在列挙しているデータのキーと[illegible]が格納されます。continueメソッドを呼ぶと、カーソルが次のデータに移動した状態で、successイベントのハンドラが再度呼ばれます。

Firefox

初期画面

「初期化」ボタンをクリックすると、データが格納され一覧で表示されます

「アップグレード」ボタンをクリックすると、データベースのバージョンが2になり、id列が追加されます

「10才から20才にフィルター」ボタンをクリックすると、年齢をもとにデータがフィルターされて表示されます

参照

indexesDB.open メソッド …………… P.387
indexesDB.deleteDataBase メソッド …… P.387
transaction メソッド ……………… P.389
objectStore メソッド ……………… P.389
close メソッド ……………………… P.387

LOCALDATA&OFFLINE.SAMPLE-03

オフライン状態とキャッシュ状態を取得する

アプリケーション・キャッシュを指定したうえで、オンライン・オフライン状態のイベントと、キャッシュ状態のイベントを取得して、その[illegible]画面に出力しています。

注意点としては、ローカル実行(「C:¥」や「file://」で始まるＵＲＬ)ではキャッシュが有効になりません。必要に応じてサーバーを用意してください。また、サーバーは、マニフェストファイルのMIMEタイプ(text/cache-manifest)に対応している必要があります。

なお、Firefoxでのみ、オフライン状態の模擬テストが可能です(「firefox」メニューの「Web開発」から「オフライン作業」を選択)。

キャッシュ・マニフェスト

```
CACHE MANIFEST
# version:2
# コメント行です。内容を変えずに更新する場合のため、バージョンを記載します。

# 既定のセクションはキャッシュ対象の指定です。
# マニフェストを指定したHTMLは、既定でキャッシュ対象となります。
sample.png

NETWORK:
# キャッシュしない対象を指定します。前方(部分)一致での指定です。
# ワイルドカード(*)も使用可能です
OfflineAPI_sample.html

FALLBACK:
# キャッシュではなく代替ファイルを使用する場合に指定します
#/online.html /offline.html

CACHE:
# 明示的にキャッシュ指定のセクションを開始します
# cache.css
```

ローカルデータとオフライン

JavaScript

```
// 要素への参照を取得します
var online_status = document.getElementById("online_status");
var output = document.getElementById("output");

// 画面上に状況を出力するための関数を定義
function write(msg) {
    output.innerHTML = msg + "¥n" + output.innerHTML;
}

// オンライン状態変更のイベント処理関数を定義
function onoffHandler(e) {
    var sts = window.navigator.onLine ? "オンライン" : "オフライン";
    online_status.innerHTML = sts;
    write("オンライン状態が変更されました:" + sts);
}

// 初期状態のオンライン状態を出力
onoffHandler();

// オンライン・オフラインイベントに登録
window.addEventListener("online", onoffHandler);
window.addEventListener("offline", onoffHandler);

// アプリケーション・キャッシュのオブジェクトを取得
var cache = window.applicationCache;

// 数値とキャッシュ状態のイベント名の対応表
var names = {
    0: "UNCACHED", 1: "IDLE", 2: "CHECKING",
    3: "DOWNLOADING", 4: "UPDATEREADY", 5: "OBSOLETE"
};

// キャッシュ状態更新時のイベント処理関数の定義
function statusHandler(e) {
    write("イベント名:" + e.type);
    write("キャッシュ状態:" + names[cache.status]);
}

// キャッシュ状態のイベント名の配列
var events = [
    "checking", "noupdate", "obsolete", "error",
    "downloading", "progress", "updateready", "cached"
];

// 上記の配列をもとにイベントに処理関数を登録
```

```
for (var i = 0; i < events.length; i++) {
    cache.addEventListener(events[i], statusHandler);
}
```

HTML

※レイアウトはCSSで指定しています

```
<body>
<div id="online_status"></div>
<img src="sample.png">
<textarea id="output"></textarea>
</body>
```

Internet Explorer

オフライン/オンラインの状態と、キャッシュの取得状況を確認できます。

Firefox

参照

キャッシュ・マニフェスト …………… P.391
applicationCache プロパティ ………… P.392
onLine プロパティ ……………………… P.393

GEOLOCATION.01

現在の位置情報を1度だけリクエストしたい

★ = window.navigator.geolocation　位置情報オブジェクトの取得

★.getCurrentPosition(◆[, ▲, ●])　現在地情報をリクエスト

★……ジオロケーション(地理位置情報)オブジェクト

◆……取得成功時のコールバック関数

▲……取得エラー時のコールバック関数　省略可能(次項参照)

●……位置取得の詳細オプション　省略可能(次項参照)

形式　プロパティ(geolocation)、メソッド(getCurrentPosition)

ユーザーの現在地を取得するには、window.navigator.geolocationに格納されているジオロケーション・オブジェクトのgetCurrentPositionメソッドを呼び出します。

メソッド自体は取得処理の起動のみで終了しますが、その後、位置情報の取得に成功すると、引数に指定した関数がPositionオブジェクトを引数にして呼び出されます。

Positionオブジェクトのtimestamp属性には、UNIXTIME(1970/1/1からのミリ秒単位の累計値)で表された取得時点が格納されます。coords属性には、以下の位置情報を持つ座標オブジェクトが格納されます。

プロパティ	解説	プロパティ	解説
latitude	緯度（単位：度数）	altidueAccuracy	高度の誤差範囲（単位：メートル）
longitude	経度（単位：度数）	heading	方角（単位：度数）
altitude	高度（単位：メートル）	speed	速度（単位：メートル毎秒）
accuracy	座標の誤差範囲（単位：メートル）		

GEOLOCATION.02

現在位置を監視し続けたい

★ = ◆.watchPosition(▲[, ●, ■]) 現在地の監視を開始

◆.clearWatch(★) 指定したIDの監視を終了

★……位置監視ID

◆……ジオロケーション(地理位置情報)オブジェクト

▲……取得成功時のコールバック関数

●……取得エラー時のコールバック関数　省略可能

■……位置取得の詳細オプション　省略可能

形式 メソッド

定期的に現在位置の変化を監視したい場合は、watchPositionメソッドを使用します。引数はgetCurrentPositionメソッドと同じです。

メソッド自体は監視IDを返してすぐに終了します。その後、現在位置の変化を監視し、変化があると引数に指定した関数がその都度呼び出されます。監視を終わらせるには、開始時にメソッドが返した監視IDを引数にしてclearWatchメソッドを呼び出します。

省略可能な第2引数には、取得失敗時に呼び出す関数を指定します。引数にはエラーオブジェクトが渡されます。エラーオブジェクトのcode属性には失敗理由がコードで格納されます(コード1は権限エラー、コード2はシステム内部エラー、コード3はタイムアウト)。

省略可能な第3引数には、位置取得の詳細なオプションを、以下の属性を持つオブジェクトの形で指定します。

プロパティ	解説
enableHighAccuracy	trueの場合、可能なら高精度な位置取得を行う
timeout	位置取得のタイムアウト時間（単位：ミリ秒）
maximumAge	位置情報の最大キャッシュ期限（単位：ミリ秒）

文例

```
var watch_id = navigator.geolocation.watchPosition(callback,
  function(err){ alert(err.message); }, { timeout:1000 });
```

詳細オプション付きで、位置情報の監視を開始しています。

ブラウザ対応表	IE10	IE9	IE8	Fx	Chrome	Safari	Opera	iOS	Android
	○	○	×	○	○	○	○	○	○

GEOLOCATION.SAMPLE-01

現在地情報を表示する

現在地の情報を取得して画面に表示させています。初回表示時にはブラウザや環境によって表示は異なりますが、位置情報の利用許可ダイアログが表示されます。

JavaScript

```
// 位置情報オブジェクトの取得
var geo = window.navigator.geolocation;

// プロパティ名と表示ラベルの対応表
var names = {latitude: "緯度:", longitude: "経度:",
    altitude: "高度:", heading: "方角:", speed: "速度:"};

if (geo) { // geolocation対応か判別

    // 現在地取得時コールバック関数
    function successHandler(pos) {
        var coords = pos.coords; // 座標オブジェクト取得
        var output = document.getElementById("output");
        for (var p in names) {
            // 座標オブジェクトの各プロパティを出力
            output.innerHTML += names[p] + coords[p] + "<br>";
        }
    }

    // 現在地取得失敗時コールバック関数
    function errHandler(err) {
        alert(err.code + ":" + err.message);
    }

    // 取得オプション定義
    var option = {enableHighAccuracy: true};

    // 現在地取得開始
    geo.getCurrentPosition(successHandler, errHandler, option);
}
```

※レイアウトはCSSで指定しています

```
<body>
<h3>現在地の位置情報です。</h3>
<div id="output"></div>
</body>
```

現在地の位置情報です。

緯度:35.689772
経度:139.7213312
高度:null
方角:null
速度:null

現在地の位置情報が表示されます。表示される情報は対応するセンサーの有無によっても異なります

geolocation プロパティ ··········· P.404
getCurrentPosition メソッド ········· P.404

LANGUAGE CORE.01

JSON形式を取り扱いたい

★ = JSON.stringify(◆) オブジェクトのJSON文字列への変換
◆ = JSON.parse(★) JSON文字列のオブジェクトへの返還

★……JSON(JavaScript Object Notation)形式の文字列
◆……オブジェクト

形式 メソッド

オブジェクトは、下記のように「{}」を使った書式で初期化することができます。

```
var obj = {name:"田中", age:24};
```

この例では、name属性が「田中」で、age属性が24になります。この書式は、入れ子にすることもできます。

JSONは、この初期化の書式をもとにオブジェクトを文字列(Unicode、既定ではUTF-8)で表現したもので、オブジェクトを保存・復元したり、送受信する場合に使用されます。

ただし、初期化の書式と異なり、JSONでは、文字列、数値、真偽値、null値という基本要素と、それらを要素とする配列またはオブジェクトのみが使用可能です。また、オブジェクトの子の階層で親のオブジェクトが参照されている場合も、無限循環になるため表現できません。

JSON形式からの復元は、eval()関数で文字列をJavaScriptコードとして解釈するだけでも可能ですが、安全性等の考慮もあり、現在では専用のメソッドが用意されています。

オブジェクトからJSON形式への変換にはJSON.stringify()メソッドを使用します。逆に、JSON形式の文字列からオブジェクトを復元するには、JSON.parse()メソッドを使用します。変換時に、オブジェクトに循環参照があればエラーになり、関数については無視されます。

文例

```
var obj =  {id:1, person:{name:"jack", age:25}};
```
オブジェクトを初期化します。

```
var json = JSON.stringify(obj);
```
オブジェクトをJSONに変換します。

▶ ブラウザ対応表	IE10	IE9	[illegible]	[illegible]	Chrome	Safari	Opera	[illegible]	Android
	○	○	○	○	○	○	○	○	○

参照

【SAMPLE】ハッシュでページ状態を切り替える・・P.414

LANGUAGE CORE.02

CSSセレクタ形式で要素を取得したい

★ = ◆.querySelector(▲)　CSSセレクタに一致する要素を取得
● = ◆.querySelectorAll(▲)　CSSセレクタに一致するすべての要素を取得

★……一致した要素
◆……documentオブジェクトまたは任意のHTML要素(element)
▲……CSSセレクタ
●……一致した要素のリスト

形式　メソッド

要素の取得には、document.getElementByIdやdocument.getElementsByClassNameなどがありますが、これらではid属性やclass属性を使った、ごく単純な指定しか行えません。

そこで、CSSで「{}」のブロックの前に記載する「CSSセレクタ」(「div#id」や「.class」などの対象指定)を使用することで、柔軟な指定を可能にしたのが、Selectors APIです。

単一の要素を取得する場合はquerySelectorメソッドの引数に、CSSセレクタを指定します。複数の要素が一致した場合は最初の要素が返ります。一致しなければnullが返ります。

複数の要素を取得する場合は、querySelectorAllメソッドを使用します。戻り値は要素のリストで、配列のようにして要素を取得できます。

どちらのメソッドも、documentオブジェクトで呼び出した場合はドキュメント全体が、特定の要素から呼び出した場合はその要素の内側にある要素が、対象範囲となります。

文例

```
var el = document.querySelector("#head, #main, #side");
```

一致した最初の要素を取得しています。

ブラウザ対応表	IE10	IE9	IE8	IE7	Chrome	Safari	Opera	iOS6	Android
	○	○	○	○	○	○	○	○	○

【SAMPLE】ハッシュでページ状態を切り替える…P.414

LANGUAGE CORE.03

ハッシュの変更イベントを取得したい

window.onhashchange = ★　　ハッシュ変更時

★……実行する命令（関数や関数名）

形式 イベント

高度なJavaScriptの機能を使用したページでは、ページ内容の更新を、サーバーからページを再取得するのではなく、JavaScriptで行いたい場合があります。

このような場合、ハッシュ（URLの「#」以降）のみが異なるURLへの移動を、ページ内容の更新のきっかけとして利用すると、ページ遷移に似せたユーザー体験を構築できます。また、この場合、画面状態とURLが対応するため、画面状態を再現するのも容易になります。

ハッシュのみの変更はwindowのhashchangeイベントで取得します。このイベントのイベント引数には、newURL属性とoldURL属性があり、変更後と変更前のURLが取得できます。

文例

```
window.onhashchange = function(event){
  if(location.hash == "first"){ // #firstに変更された場合
    doFirst(event.oldURL);
  } else if(location.hash == "second"){ // #secondに変更された場合
    doSecond(event.oldURL);
  }
}
```

ハッシュの変更に応じて関数を呼び出しています。

参照 【SAMPLE】ハッシュでページ状態を切り替える･･･P.414

LANGUAGE CORE.SAMPLE-01

ハッシュでページ状態を切り替える

同じページ内でハッシュだけを切り替えるリンクを用意し、各リンクをクリックすることでハッシュが切り替わるのに応じて(hashchangeイベント)、画面の文字色、背景色などを更新しています。

各ハッシュでの画面状態は、キーと値の形式で、専用のオブジェクトに保持しています。画面が切り替わるごとに、現在の画面状態の設定を、このオブジェクトをJSON形式に変換して画面表示しています。

要素の取得にはCSSセレクタを使用しています。

JavaScript

```
// 各ページ状態の設定を格納したオブジェクト
var settings = {
    first: {title: 'First',
        style: 'color:black;background-color:white;'},
    second: {title: 'Second',
        style: 'color:green;background-color:black;'},
    third: {title: 'Third',
        style: 'color:blue;background-color:black;'}
};

// 現在のハッシュによって画面を切り替えるための関数
function changeStatus() {
    var current;
    // ハッシュによって画面設定を取得
    switch (location.hash) {
        case "#first":
            current = settings.first;
            break;
        case "#second":
            current = settings.second;
            break;
        case "#third":
            current = settings.third;
            break;
        default:
            current = settings.first;
```

```
            break;
    }

    // CSSセレクタで要素を取得
    var title = document.querySelector("h1#title");
    var body = document.querySelector("body");
    var output = document.querySelector("div#output");

    // 画面設定オブジェクトのデータを要素に適用
    title.innerHTML = current.title;
    output.innerHTML = JSON.stringify(current);
    body.setAttribute("style", current.style);

}
;

// 初回読み込み時にハッシュをチェック
window.onload = changeStatus;

// ハッシュ変更時にも画面更新
window.onhashchange = changeStatus;
```

HTML

※レイアウトはCSSで指定しています

```
<body>
<h1 id="title">First</h1>
<div>
    <ul>
        <li><a href="#first">first</a></li>
        <li><a href="#second">second</a></li>
        <li><a href="#third">third</a></li>
    </ul>
</div>
<h3>現在のページ設定(JSON書式)</h3>
<div id="output"></div>
</body>
```

Internet Explorer

First

- first
- second
- third

現在のページ設定（JSON書式）

{"title":"First","style":"color:black;background-color:white;"}

初期状態ではこのような画面です

secondをクリックするとこう変化します

JSON.stringify メソッド ･･････････････ P.410
querySelector メソッド ･･････････････ P.412
onhashchange イベント ･･････････････ P.413

第3部
オブジェクト一覧
OBJECT LIST

- ビルトインオブジェクトとナビゲーターオブジェクト
- DOM
- XMLHttpRequestオブジェクト

OBJECT LIST.01

ビルトインオブジェクトとナビゲーターオブジェクト

JavaScriptのオブジェクトとそのメソッドおよびプロパティの一覧です。それぞれのオブジェクトには決められたメソッドとプロパティが用意されています(詳しくはp.014を参照)。また、どのオブジェクトにも属さない処理を実行するビルトイン関数と呼ばれる命令もあります。

ビルトイン関数(オブジェクトに属さない関数)

関数		
escape()	文字列をエスケープ文字にエンコードする	250
eval()	数式や文字列をJavaScriptの命令として実行する	249
isFinite()	値がevalで評価できる有限な数ならtrue、それ以外ならfalseを返す	
isNaN()	値が数値でない場合はtrue、それ以外ならfalseを返す	251
Number()	値を数値に変換する。変換できない場合はNaNを返す	
parseFloat()	文字列を浮動小数点数に変換する	247
parseInt()	文字列を指定の進数から[illegible]に変換する	247
String()	指定された値を文字列に変換する	
unescape()	エンコードされた文字列をデコードして、元の文字列に戻する	250
void()	何も値を返さない	246

イベントハンドラ　128

イベント		
onabort	画像の読み込み中断時に発生する	130
onblur	フォームエレメントやウィンドウからフォーカスが外れたときに発生する	132
onchange	フォームの部品(入力欄の文字列やメニューで選択される項目)の状態が変化したときに発生する	138
onclick	マウスが左クリックされ、離されたときに発生する	134
oncontextmenu	マウスの右ボタンがクリックされたときに発生する	135
ondblclick	マウスがダブルクリックされたときに発生する	134
onerror	画像の読み込み失敗時に発生する	130
onfocus	フォーカスが合ったときに発生する	132
onkeydown	キーが押されたときに発生する	139
onkeypress	キーが押されている間、断続的に発生する	139
onkeyup	キーが押されて離されたときに発生する	139
onload	ページの読み込み完了時に発生する	128
onmousedown	マウスの左ボタンがクリックされたときに発生する	134
onmouseout	マウスカーソルが外れたときに発生する	133
onmouseover	マウスカーソルが重なったときに発生する	133
onmouseup	マウスのボタンが離されたときに発生する	134
onreset	resetボタンが押されたときに発生する	136

onresize	オブジェクトのサイズ変更時に発生する	131
onselect	入力フィールド選択時に発生する	138
onsubmit	submit(送信)ボタンが押されたときに発生する	136
onunload	ページの切り替え時に発生する	128

Anchorオブジェクト 230

document.anchors

プロパティ

length	ドキュメント中のアンカー数を参照する	230

Appletオブジェクト

document.applets

プロパティ

length	ドキュメント中のアンカー数を参照する	

Arrayオブジェクト 168

プロパティ

length	配列の要素数を参照／設定する	168

メソッド

concat()	2つの配列を連結する	174
join()	配列の要素を指定した区切り文字で連結する	174
pop()	配列の最後尾の要素を取り出す	170
push()	配列の最後尾に要素を追加する	170
reverse()	配列要素の並び順を反転させる	172
shift()	配列の先頭の要素を取り出す	170
slice()	指定した範囲の配列の要素を取り出す	174
sort()	配列要素を並べ替える	172
splice()	配列の要素を置換する	174
unshift()	配列の先頭に要素を追加する	170

Booleanオブジェクト 252

共通のメソッド・プロパティ(p.427)を参照。

Dateオブジェクト 180

メソッド

getDate()	日を返す(1～31)	182
getDay()	曜日を返す(0:日～6:土)	182
getFullYear()	4桁の西暦を返す	182
getHours()	時を返す(0～23)	185
getMilliseconds()	ミリ秒(1／1000秒)を返す(0～999)	185
getMinutes()	分を返す(0～59)	185
getMonth()	月を返す(0～11)	185
getSeconds()	秒を返す(0～59)	185
getTime()	1970年1月1日午前0時からの経過秒数をミリ秒で返す	186
getTimezoneOffset()	協定世界時との時差で返す	190
getUTCDate()	協定世界時の日で返す	190
getUTCDay()	協定世界時の曜日で返す	190
getUTCFullYear()	協定世界時の4桁の西暦で返す	190
getUTCHours()	協定世界時の時で返す	190
getUTCMilliseconds()	協定世界時のミリ秒で返す	190

getUTCMinutes()	協定世界時の分で返す	190
getUTCMonth()	協定世界時の月で返す	190
getUTCSeconds()	協定世界時の秒で返す	190
getYear()	西暦を返す	182
parse()	1970年1月1日午前0時からの経過秒数をミリ秒で返す	186
setDate()	日を設定する	181
setFullYear()	4桁の西暦を設定する	181
setHours()	時を設定する	184
setMilliseconds()	ミリ秒を設定する	184
setMinutes()	分を設定する	184
setMonth()	月を設定する	181
setSeconds()	秒を設定する	184
setTime()	1970年1月1日午前0時からの経過秒数をミリ秒で設定する	
setUTCDate()	協定世界時の日で設定する	189
setUTCFullYear()	協定世界時の4桁の西暦で設定する	189
setUTCHours()	協定世界時の時で設定する	189
setUTCMilliseconds()	協定世界時のミリ秒で設定する	189
setUTCMinutes()	協定世界時の分で設定する	189
setUTCMonth()	協定世界時の月で設定する	189
setUTCSeconds()	協定世界時の秒で設定する	189
setYear()	西暦を設定する	181
toGMTString()	グリニッジ[illegible]で返す	188
toLocaleString()	ローカル時で返す	188
toUTCString()	協定世界時で返す	188
UTC()	1970年1月1日午前0時からの[illegible]秒数をミリ秒で返す	186

Documentオブジェクト 056

プロパティ

alinkColor	リンクを[illegible]した[illegible]の文字色を参照／設定する	
bgColor	背景色を参照／設定する	
cookie	クッキーの文字列を参照／設定する	064
domain	ドメイン名を参照する	060
fgColor	文字色を参照／設定する	
lastModified	[illegible]更新日を参照する	059
linkColor	リンクの文字色を参照／設定する	
location	ドキュメントのURIを参照／設定する	226
referrer	リンク元のURIを参照する	238
title	ドキュメントのタイトルを参照する	061
URL	現在のページのURIを参照する	226
vlinkColor	訪問済みリンクの文字色を参照／設定する	

メソッド

clear()	ドキュメントの内[illegible]を消去する	
close()	openメソッドで開始したドキュメントの出力を終了する	057
getSelection()	選択された文字を取得する	062
open()	ドキュメントの出力を開始する	057
write()	データを書き出す	058
writeln()	データを書き出して改行する	058

Elementオブジェクト 104

Buttonオブジェクト　<input type="button">タグで作成されるボタン

プロパティ

name	エレメントの名前を参照する	108
type	エレメントの種類を参照する	108
value	エレメントの値を参照／設定する	114

メソッド

blur()	フォーカスを外す	115
click()	自動的にクリックする	115
focus()	フォーカスを合わせる	115

Checkboxオブジェクト　<input type="checkbox">タグで作成されるチェックボタン

プロパティ

checked	チェック状態を参照／設定する。チェックされている場合はtrue、されていない場合はfalseを返す	111
defaultChecked	初期チェック状態を参照する。チェックされている場合はtrue、されていない場合はfalseを返す	113
name	エレメントの名前を参照する	108
type	エレメントの種類を参照する	108
value	エレメントの値を参照／設定する	114

メソッド

blur()	フォーカスを外す	115
click()	自動的にクリックする	115
focus()	フォーカスを合わせる	115

FileUploadオブジェクト　<inputtype="file">タグで作成されるファイルアップロードのフィールド

プロパティ

name	エレメントの名前を参照する	108
type	エレメントの種類を参照する	108
value	エレメントの値を参照／設定する	114

メソッド

blur()	フォーカスを外す	115
focus()	フォーカスを合わせる	115
select()	フィールドを選択状態にする	115

Hiddenオブジェクト　<input type="hidden">タグで作成される隠しフィールド

プロパティ

name	エレメントの名前を参照する	108
type	エレメントの種類を参照する	108
value	エレメントの値を参照／設定する	114

Optionオブジェクト　<option>タグで作成される選択メニューの選択肢

プロパティ

defaultSelected	初期選択状態を参照する。選択されている場合はtrue、されていない場合はfalseを返す	113
index	選択項目の参照番号を参照する	
selected	選択状態を参照する。選択されている場合はtrue、されていない場合はfalseを返す	111
text	選択メニューの[illegible]を参照／設定する	114
value	エレメントの値を参照／設定する	114

メソッド		
blur()	フォーカスを外す	115
focus()	フォーカスを合わせる	115
select()	文字を選択状態にする	115

Radioオブジェクト	<input type="radio">で作成されるラジオボタン	
プロパティ		
checked	チェック状態を参照／設定する。チェックされている場合はtrue、されていない場合はfalseを返す	111
defaultChecked	初期チェック状態を参照する。チェックされている場合はtrue、されていない場合はfalseを返す	113
name	エレメントの名前を参照する	108
type	エレメントの種類を参照する	108
value	エレメントの値を参照／設定する	114
メソッド		
blur()	フォーカスを外す	115
click()	自動的にクリックする	115
focus()	フォーカスを合わせる	115

Resetオブジェクト	<input type="reset">タグで作成されるリセットボタン	
プロパティ		
name	エレメントの名前を参照する	108
type	エレメントの種類を参照する	108
value	エレメントの値を参照／設定する	114
メソッド		
blur()	フォーカスを外す	115
click()	自動的にクリックする	115
focus()	フォーカスを合わせる	115

Selectオブジェクト	<select>タグで作成されるプルダウンまたはリストボックス	
プロパティ		
length	選択項目の数を参照する	108
name	エレメントの名前を参照する	108
options	選択状態を参照する	
selectedIndex	選択されている項目の要素番号を参照する	112
type	エレメントの種類を参照する	108
value	エレメントの文字列を参照／設定する	114
メソッド		
blur()	フォーカスを外す	115
focus()	フォーカスを合わせる	115

Submitオブジェクト	<input type="submit">タグで作成される送信ボタン	
プロパティ		
name	エレメントの[illegible]を参照する	108
type	エレメントの名前を参照する	108
value	エレメントの値を参照／設定する	114
メソッド		
blur()	フォーカスを外す	115
click()	自動的にクリックする	115
focus()	フォーカスを合わせる	115

Textオブジェクト	<input type="text">タグで作成される入力フィールド	
プロパティ		
defaultValue	初期文字列を参照する	114
name	エレメントの名前を参照する	108
type	エレメントの[illegible]を参照する	108
value	エレメントの値を参照／設定する	114
メソッド		
blur()	フォーカスを外す	115
click()	自動的にクリックする	115
focus()	フォーカスを合わせる	115
select()	文字を[illegible]状態にする	115

Textareaオブジェクト	<textarea>タグで作成される複数行の入力フィールド	
プロパティ		
defaultValue	初期文字列を参照する	114
name	エレメントの名前を参照する	108
type	エレメントの種類を参照する	108
value	エレメントの値を参照／設定する	114
メソッド		
blur()	フォーカスを外す	115
focus()	フォーカスを合わせる	115
select()	文字を[illegible]状態にする	115

[illegible]オブジェクト		
document.embeds		
プロパティ		
length	ドキュメント中のプラグイン[illegible]を参照する	065

[illegible]		[illegible]
プロパティ		
clientX	表示領域上のマウスのｘ座標を参照／[illegible]する	142
clientY	表示領域上のマウスのｙ座標を参照／設定する	142
keycode	入力されたキーのキーコード（文字コード）を参照する	140
layerX	イベントが発生したレイヤ上のX[illegible]を[illegible]す	
layerY	イベントが発生したレイヤ上のY[illegible]を[illegible]す	
pageX	イベントが発生したページ上のX[illegible]を返す	142
pageY	イベントが発生したページ上のY[illegible]を返す	142
screenX	イベントが発生した画面上のX座[illegible]を返す	142
screenY	イベントが発生した画面上のY[illegible]を返す	142
target	イベントの発生元となるオブジェクトを返す	141
type	イベントの種類を参照する	141
x	マウスのx座標を参照する	142
y	マウスのy[illegible]標を参照する	142

Formオブジェクト　104

プロパティ

action	フォームの送信先を参照／設定する	106
encoding	フォーム送信時のエンコード方式を参照／設定する	106
length	ドキュメント中のフォームの数、フォーム中のエレメントの[illegible]を参照する	104
method	フォームの送信形式を参照／設定する	106
name	フォームの名前を参照する	104
target	フォーム送信後のターゲットウィンドウを参照／設定する	106

メソッド

reset()	フォーム内容をリセットする	110
submit()	フォーム内容を送信する	110

Frameオブジェクト

プロパティ

location	フレームのURIを参照する
name	フレーム名を[illegible]／設定する
parent	親フレームを参照する
self	自分自身のフレームを参照する
top	最上位のフレームを参照する

window.frames

プロパティ

length	ドキュメント中のフレームの総数を参照する

※ほか、Windowオブジェクトのプロパティやメソッドを使用することが可能

Functionオブジェクト　280

プロパティ

arguments	関数に渡される引数を参照する	281
arity	[illegible]に渡される引数の数を参照する	281
caller	呼び出し元のスクリプトの内容を参照する	281

メソッド

apply()	[illegible]内からほかの関数を[illegible]び出す。任意の数の引数を指定できる	282
call()	関数内からほかの関数を呼び出す	282

Historyオブジェクト　238

プロパティ

length	履歴の数を参照する	239

メソッド

back()	1つ前のページに戻る	240
forward()	1つ後のページに進む	240
go()	指定した数だけ前後の履歴に移動する	240

Imageオブジェクト 220

プロパティ

border	画像のボーダーの太さを参照／設定する	
complete	画像の読み込みが完了したか参照し、画像が完全に読み込まれている場合はtrue、読み込まれていない場合はfalseを返す	224
height	画像の高さを参照／設定する	222
hspace	テキストと画像の左右の間隔を参照／設定する	
lowsrc	低解像度用画像のURIを参照／設定する	223
name	画像の名前を参照する	222
src	画像のURIを参照／設定する	223
vspace	テキストと画像の上下の間隔を参照／設定する	
width	画像の幅を参照／設定する	222

document.images

プロパティ

length	ドキュメント中の画像の数を参照／設定する	220

JSONオブジェクト 410

メソッド

stringify()	オブジェクトをJSON文字列に変換する	410
parse()	JSON文字列をオブジェクトに変換する	410

Linkオブジェクト・Areaオブジェクト 228

プロパティ

hash	リンク先のアンカーを参照／設定する	
host	リンク先のホスト情報を返す	231
hostname	リンク先のパス名を参照する	231
href	リンク先または現在のページのURIを参照／設定する	228
pathname	リンク先のページのパス名を参照する	231
port	リンク先のポート番号を参照する	231
protocol	リンク先のプロトコルを参照する	231
search	CGIなどに渡されるサーチ部分を参照／設定する	
target	ターゲットウィンドウを参照／設定する	229

document.links

プロパティ

length	ドキュメント中のリンク数を参照する	228

※lengthプロパティはLinkオブジェクトのみ

Locationオブジェクト 226

プロパティ

hash	現在のアンカーを参照、または指定したアンカーへ移動する	
host	指定したページのホスト情報を返す	231
hostname	指定したページのホスト名を参照する	231
href	指定したページのURIを参照／設定する	228
pathname	指定したページのパス名を参照する	231
port	指定したページのポート番号を参照する	231
protocol	指定したページのプロトコルを参照する	231
search	CGIなどに渡されるサーチ部分を参照／設定する	

メソッド

reload()	ページをリロード（再読み込み）する	227
replace()	ページのURIを変更し、ページを移動する	232

メソッド

javaEnabled()	Javaが使用可能な場合はtrue、使用不可の場合はfalseを返す	212

Numberオブジェクト

プロパティ

MAX_VALUE	JavaScriptで使用可能な最大値を参照する	266
MIN_VALUE	JavaScriptで使用可能な最小値を参照する	266
NaN	数値以外であることを表す	266
NEGATIVE_INFINITY	負の無限大を参照する	266
POSITIVE_INFINITY	正の無限大を参照する	266

Objectオブジェクト(共通のメソッド・プロパティ) 272

プロパティ

constructor	オブジェクトを作成した関数を参照する	273
prototype	オブジェクトにプロパティやメソッドを追加する	

※constructorプロパティとprototypeプロパティは、Array、Boolean、Date、Function、Number、Object、RegExp、Stringオブジェクトが持つプロパティ

メソッド

toSource()	指定した関数やオブジェクトの内容を返す	274
toString()	数値をn進数表記の文字列に変換する	248
unwatch()	指定したプロパティの監視を中止する	
watch()	指定したプロパティを監視する	
valueOf()	オブジェクトの値を返す	273

Pluginオブジェクト 213

プロパティ

description	プラグインの詳細情報を参照する	213
filename	プラグインのファイル名を参照する	213
name	プラグインの名前を参照する	213

document.plugins/navigator.plugins

プロパティ

length	プラグイン数を参照する	213

RegExpオブジェクト 286

プロパティ

$1,...,$9	一致した文字列を参照する	
global	完全一致を検索する場合はtrue、検索しない場合falseを返す	288
ignoreCase	大文字、小文字を区別する場合true、区別しない場合falseを返す	288
lastIndex	検索の開始位置を参照/設定する	291
lastMatch	最後に一致した文字列を参照する($&でも可)	289
lastParen	最後に一致したグループの文字列を参照する($+でも可)	289
leftContext	最後に一致した文字列より前の文字列を参照する($`でも可)	290
multiline	改行コードを無視するかどうかを参照/設定する($*でも可)	288
rightContext	最後に一致した文字列より後ろの文字列を参照する($'でも可)	290
source	パターン文字列を参照する	291

メソッド

compile()	パターン文字列を設定/変更する	291
exec()	検索を実行する	291
input()	検索する文字列を参照/設定する	291

test()	一致する文字列が含まれているかどうかを調べ、あった場合はtrue、なかった場合はfalseを返す	291

Screenオブジェクト 098

プロパティ

availHeight	有効な領域の高さを参照する	098
availWidth	有効な領域の幅を参照する	098
availLeft	有効な左端のX座[illegible]を参照する	098
availTop	有効な上端のY[illegible]を参照する	098
colorDepth	表示できる色[illegible]を参照する	101
height	モニタの高さを参照する	100
pixelDepth	オフスクリーンの色深度を参照する	101
width	モニタの[illegible]を参照する	100

Stringオブジェクト 198

プロパティ

length	文字列の長さを参照する	198

メソッド

anchor()	文字列にアンカー名を設定する	199
big()	文字列を大きくする	
blink()	文字列を点滅させる	
bold()	文字列を太字にする	
charAt()	指定した[illegible]の文字を抜き出す	204
charCodeAt()	指定した位置の文字をUnicodeの値に変換する	203
concat()	文字列を結合する	205
fixed()	文字列を等幅フォントにする	
fontcolor()	文字列の色を設定する	
fontsize()	文字列のサイズを設定する	
fromCharCode()	Unicodeの値を文字に[illegible]	203
indexOf()	文字列を検索する	202
italics()	文字列を斜体にする	
lastIndexOf()	文字列を後ろから検索する	202
link()	文字列にリンクを設定する	199
match()	検索し、一致した文字列を返す	293
replace()	文字列中の指定した文字列を[illegible]る	293
search()	文字列の検索を行い、一致した位置を返す	293
slice()	指定した[illegible]の文字列を[illegible]き出す	205
small()	文字列を小さくする	
split()	文字列を区切り文字で分割し、配列として返す	201
strike()	文字列を打消し線付きにする	
sub()	文字列を下付き文字にする	
substr()	指定した位置から指定した文字数分の文字列を[illegible]き出す	205
substring()	指定した[illegible]の文字列を抜き出す	205
sup()	文字列を上付き文字にする	
toLowerCase()	アルファベットを小文字に変換する	200
toUpperCase()	アルファベットを大文字に変換する	200

windowオブジェクト 074

プロパティ

closed	ウィンドウが閉じているかを参照し、閉じられていた場合はtrue、開いていた場合はfalseを返す	077
defaultStatus	ステータスバーのデフォルトの文字列を参照／設定する	
innerHeight	ウィンドウの内側(表示領域)の高さを参照／設定する	080
innerWidth	ウィンドウの内側(表示領域)の幅を参照／設定する	080
length	ウィンドウ内にあるフレームの総数を参照する	
name	ウィンドウ名を参照／設定する	077
opener	現在のウィンドウを開いた元のウィンドウを参照する	077
outerHeight	ウィンドウの外側(ウィンドウ全体)の高さを参照／設定する	080
outerWidth	ウィンドウの外側(ウィンドウ全体)の幅を参照／設定する	080
pageXOffset	横方向のオフセットを参照／設定する	082
pageYOffset	縦方向のオフセットを参照／設定する	082
status	ステータスバーの文字列を参照／設定する	

メソッド

alert()	警告ダイアログを表示する	050
back()	1つ前のページに戻る	083
blur()	ウィンドウからフォーカスを外す	132
clearInterval()	SetIntervalメソッドで設定したタイマーを解除する	165
clearTimeout()	SetTimeoutメソッドで設定したタイマーを解除する	165
close()	ウィンドウを閉じる	057
confirm()	確認ダイアログを表示し、[OK]ボタンでtrue、それ以外の場合はfalseを返す	051
find()	指定した文字列を検索する	083
focus()	ウィンドウにフォーカスを合わせる	132
forward()	1つ先のページに進む	083
home()	ホームページに移動する	083
moveBy()	ウィンドウの表示位置を指定した距離だけ移動させる	078
moveTo()	ウィンドウの表示位置を指定した座標に移動させる	078
open()	新しいウィンドウを開く	057
print()	印刷する	083
prompt()	文字入力ダイアログを表示し、[OK]ボタンで入力された文字列、それ以外ではnullを返す	052
resizeBy()	ウィンドウのサイズを指定した分だけ現在のサイズから変更する	079
resizeTo()	ウィンドウのサイズを指定した幅と高さに変更する	079
scroll()	ページの内容の表示開始位置を指定した座標まで移動させる	081
scrollBy()	ページの内容の表示開始位置を指定した距離だけ移動させる	081
scrollTo()	ページの内容の表示開始位置を指定した座標まで移動させる	081
setInterval()	一定時間ごとに関数を呼び出すタイマーを設定する	165
setTimeout()	一定時間後に関数を呼び出すタイマーを設定する	164
stop()	読み込みを中止する	083

OBJECT LIST.02

DOM

DOM(Document Object Model)のメソッドおよびプロパティです。ここで紹介するのは本書で扱ったものに限定しています。

DOM　300

プロパティ

attributes	属性のリストを参照する	310
childNodes	子ノードのリストを参照する	302
firstChild	最初の子ノードを参照する	302
innerHTML	要素のHTML/XHTMLタグと内容を参照／設定する	308
innerText	ノードの文字列を参照／設定する	308
lastChild	最後の子ノードを参照する	302
nextSibiling	次のノードを参照する	302
nodeName	ノード名を参照する	308
nodeType	ノードの種類を参照する	308
nodeValue	ノードの値を参照／設定する	308
parentNode	親ノードを参照する	302
previousSibiling	前のノードを参照する	302
tagName	要素名を参照する	308
textContent	ノードの文字列を参照／設定する	308

メソッド

appendChild()	子ノードをノードの末尾に追加する	306
cloneNode()	ノードを複製する	306
createAttribute()	[illegible]を作成する	312
createElement()	要素ノードを作成する	304
createTextNode()	テキストノードを作成する	304
getAttribute()	属性の値を取得する	310
getAttributeNode()	属性ノードを取得する	310
getElementById()	IDを持つ属性ノードを取得する	300
getElementsByName()	指定したname属性値を持つ要素を取り出し、配列として返す	300
getElementsByTagName()	指定した要素名の要素を取り出し、配列として返す	300
hasAttribute()	指定した属性がある場合はtrue、ない場合はfalseを返す	310
hasAttributes()	属性がある場合はtrue、ない場合はfalseを返す	310
hasChildNodes()	子ノードがある場合はtureを、ない場合はfalseを返す	302
insertBefore()	特定の位置に子ノードを追加する	306

item()	参照番号のノードを取得する	302
querySelector()	CSSセレクタに一致する要素を取得する	412
querySelectorAll()	CSSセレクタに一致するすべての要素を取得する	412
removeAttribute()	要素から[illegible]を削除する	314
removeAttributeNode()	属性ノードを削除する	314
removeChild()	子ノードを削除する	305
replaceChild()	子ノードを置換する	305
setAttribute()	属性と値を追加する	312
setAttributeNode()	属性ノードを追加する	312

XMLHttpRequestオブジェクト

HTTPプロトコルを使って非同期通信／同期通信を行うXMLHttpRequestオブジェクトのメソッドおよびプロパティの一覧です。なお、Internet Explorer 7より前のバージョンではXMLHTTPを利用しますが(p.333)、持っているプロパティやメソッドはほぼ同じです。

付録
APPENDIX

- スタイルプロパティ一覧
- JavaScriptインデックス
- 用語インデックス

APPENDIX.01

スタイルプロパティ一覧

JavaScriptで利用されるスタイルプロパティの一覧です。プロパティの名前や値は基本的にCSSのプロパティと対応しているため、一覧でもCSSのプロパティを併記しました。

[illegible]プロパティ	スタイルプロパティ	説明	値[illegible]指定方[illegible]
テキスト			
letter-spacing	letterSpacing	文字[illegible]	normal 実数値+単位
line-height	lineHeight	行の高さ	normal 実数値+単位 実数値 パーセント値+%
text-align	textAlign	行揃え	left right center justify
text-decoration	textDecoration	テキストの[illegible]	none underline overline line-through blink
text-indent	textIndent	インデント	実数値+単位 パーセント値+%
text-transform	textTransform	単語の表記方法	none capitalize uppercase lowercase
vertical-align	verticalAlign	文字の垂直位置	baseline top middle bottom sub super
white-space	whiteSpace	空白処理	normal nowrap

CSSのプロパティ	スタイルプロパティ	説明	値の指定方法
			pre
word-spacing	wordSpacing	単語間隔	normal 実数値+単位
フォント			
font	font	フォントのプロパティの一括指定	font-style font-variant font-weight font-size/line-height font-familyの各値
font-family	fontFamily	使用するフォント	フォントファミリー名 総称ファミリー名(serif、sans-serif、cursive、fantasy、monospace)
font-size	fontSize	フォントサイズ	xx-small、x-small、small、medium、large、x-large、xx-large larger,smaller 実数値+単位 パーセント値+%
font-style	fontStyle	斜体	italic oblique normal
font-variant	fontVariant	スモールキャップ	normal small-caps
font-weight	fontWeight	フォントの太さ	数値(100、200、300、400、500、600、700、800、900) normal bold bolder,lighter
色と背景			
background	background	背景のプロパティの一括指定	background-color background-image background-repeat background-attachment background-positionの各値
background-attachment	backgroundAttachment	背景画像の固定	scroll fixed
background-color	backgroundColor	背景色	色 transparent
background-image	backgroundImage	背景画像	URI none
background-position	backgroundPosition	背景画像の位置	実数値+単位

CSSのプロパティ	スタイルプロパティ	説明	値の指定[illegible]
			パーセント値+%
			left,center,right/ top,center,bottom
background-repeat	backgroundRepeat	背景画像の繰り返し方法	repeat
			repea t -x
			repeat-y
			no-repeat
color	color	文字色	色
ボックス			
border	border	ボーダーの一括指定	border-width border-style border-colorの各値
border-color	borderColor	ボーダーの色の一括指定	[illegible]
			transparent
border-style	borderStyle	ボーダーの種類の一括指定	none
			hidden
			dotted
			dashed
			solid
			double
			groove
			ridge
			inset
			outset
border-top	borderTop	ボーダーごとの プロパティの一括指定	border-width border-style border-color の[illegible]
border-right	borderRight		
border-bottom	borderBottom		
border-left	borderLeft		
border-top-color	borderTopColor	上下左右のボーダーの色	[illegible]
border-right-color	borderRightColor		transparent
border-bottom-color	borderBottomColor		
border-left-color	borderLeftColor		
border-top-style	borderTopStyle	上下左右のボーダーの種類	none
border-right-style	borderRightStyle		hidden
border-bottom-style	borderBottomStyle		dotted
border-left-style	borderLeftStyle		dashed
			solid
			double
			groove

CSSのプロパティ	スタイルプロパティ	説明	値の指定方法
			ridge
			inset
			outset
border-top-width	borderTopWidth	上下左右のボーダーの幅	thin
border-right-width	borderRightWidth		medium
border-bottom-width	borderBottomWidth		thick
border-left-width	borderLeftWidth		実数値+単位
border-width	borderWidth	ボーダーの[illegible]の一括指定	thin
			medium
			thick
			実数値+[illegible]
height	height	内容領域の高さ	[illegible]+単位
			パーセント値+%
			auto
margin	margin	マージンの一括指定	実数値+単位
			パーセント値+%
			auto
margin-top	marginTop	上下左右のマージン	実数値+単位
margin-right	marginRight		パーセント値+%
margin-bottom	marginBottom		auto
margin-left	marginLeft		
padding	padding	パディングの一括指定	[illegible]+単位
			パーセント値+%
padding-top	paddingTop	上下左右のパディング	[illegible]+[illegible]
padding-right	paddingRight		パーセント値+%
padding-bottom	paddingBottom		
padding-left	paddingLeft		
width	width	内[illegible]領域の幅	[illegible]+単位
			パーセント値+%
			auto

配[illegible]と表示			
float	cssFloat／styleFloat(IEのみ)	浮動(フロート)と回り込み	left
			right
			none
clear	clear	回り込みの解除	left
			right
			both
			none
display	display	表示形式	inline

CSSのプロパティ	スタイルプロパティ	説明	値の指定[illegible]
			block
			list-item
			marker
			none
			run-in、compact
			table、inline-table、table-row-group、table-header-group、table-footer-group、table-row、table-column-group、table-column 、table-cell、table-caption
overflow	overflow	はみだした内容の表示方法	visible
			hidden
			scroll
			auto
position	position	配置方法	static
			relative
			absolute
			fixed
top	top	基準となるボックスから該当要素のボックスまでの距離	実数値＋[illegible]
right	right		パーセント値＋%
bottom	bottom		auto
left	left		
z-index	zIndex	要素の重なる順序	整数値
visibility	visibility	ボックスの表示・非表示	visible
			hidden
			collapse
			リスト
リストとテーブル			
list-style	listStyle	リストのマーカーの一括指定	list-style-type
			list-style-image
			list-style-positionの各値
list-style-image	listStyleImage	リストのマーカー[illegible]	URI
			none
list-style-position	listStylePosition	リストのマーカーの配置	outside
			inside

CSSプロパティ	スタイルプロパティ	説明	値の指定方法
list-style-type	listStyleType	リストのマーカー	disc circle square decimal decimal-leading-zero lower-roman upper-roman lower-greek lower-alpha lower-latin upper-alpha upper-latin hebrew armenian georgian cjk-ideographic hiragana katakana hiragana-iroha katakana-iroha none
border-spacing	borderSpacing	セルのボーダーの間隔	[illegible]+単位
border-collapse	borderCollapse	セルのボーダーの表示形式	collapse separate
caption-side	captionSide	キャプションの[illegible]	top bottom left right
empty-cells	emptyCells	空セルのボーダーの表示・非表示	show hide
table-layout	tableLayout	表の表示方法	fixed auto

JavaScriptインデックス

H

I

J

K

L

M

APPENDIX.03

用語インデックス

記号

英数字

た

な

は

ま

や

ら

■ Information

翔泳社のWeb辞典シリーズのホームページでは、本書のサンプルデータダウンロードのほか、カラーチャートや正誤表を掲載しています。
ぜひご利用ください。

サンプルデータはこちら
http://www.shoeisha.com/book/pc/dic/

なお、スマートフォンからサンプルデータを閲覧する際には、右のQRコードからアクセスいただけます。

■ Author

株式会社アンク　http://www.ank.co.jp/

■ Staff

装丁	米倉 英弘（株式会社 細山田デザイン事務所）
本文デザイン	尾花 暁
DTP	株式会社エストール

JavaScript辞典 第4版
［HTML5対応］

2013年6月13日　初版第1刷発行

著　者	(株)アンク
発行人	佐々木 幹夫
発行所	株式会社 翔泳社（http://www.shoeisha.co.jp）
印刷・製本	大日本印刷株式会社

ISBN978-4-7981-3160-3　　Printed in JAPAN

CONTENTS

9784798131603

1923055023000

ISBN978-4-7981-3160-3
C3055 ¥2300E

株式会社翔泳社
定価：本体2,300円+税

JavaScript辞典 第4版

(株)アンク 著

HTML5対応